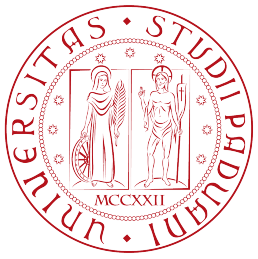


Università degli Studi di Padova

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Ph.D. course in Physycs

Doctoral Thesis

Innovative readout architecture for Monolithic Active Pixel Sensors

Ph.D. school coordinator:
Prof. Giulio Monaco
Supervisor:
Prof. Piero Giubilato

Ph.D. student:
Davide Chiappara
1229775

XXXV cycle

Abstract

Particle tracking plays an important role in many fields, from fundamental High Energy Physics research performed in the experimental halls at CERN, to the latest medical imaging equipment for the treatment of cancer; from industrial cameras used to spot manufacturing defects, to space telescopes used to study the universe. In the technology of particle tracking, the leading role belongs to silicon sensors, which over the last decades have improved performance with respect to all key metrics: high spatial and timing resolution, low power consumption, low material budget, decreasing production and exercise costs, and high adaptability to several tracking contexts. In particular, Monolithic Active Pixel Sensors (MAPS), which embed in the same silicon die both the sensor and the signal handling electronics, are the latest incarnation of the silicon sensor paradigm, and they show great potentialities to further advance the state-of-the-art in particle tracking.

Currently there are many MAPS developments in the worldwide scientific and industrial communities. The ARCADIA collaboration of the National Institute for Nuclear Physics (INFN) aims to develop a MAPS tailored towards large-area and low-power tracking applications. The ARCADIA sensor design is inspired by the ALPIDE sensor, a MAPS developed by the CERN ALICE collaboration which is the current benchmark for monolithic tracking pixel detectors.

This thesis first briefly introduces the overall development of ARCADIA sensor, and then it details the implementation of the experiment-driven readout, describes validation tests, and reports on a preliminary campaign of displacement damage dose effects performed to verify the sensor durability in a harsh environment.

Moreover, a very innovative readout architecture relying on an alternative approach for MAPS readout is discussed, with the intent of developing a sensor suitable for extreme-low power applications (such as for outer space applications). The idea takes advantage of several projections on different axes to reconstruct the addresses of the hits; this approach is made possible by data sparsification in the targeted applications. This architecture is ready at a digital electronics level and its performance has been evaluated both with a mathematical approach, using a specifically developed formalism, and with Monte Carlo simulations reproducing potential applications. Simulations were developed using both a hardware-level language as testbenches for validation, and a software framework capable of providing higher statistics. Preliminary calculations on the power consumption and on the timing performances of the architecture are mentioned. Tentative silicon runs are planned in the near future.

The immediate scope of this work is to contribute to the effort to validate the ARCADIA device, currently under development. However the larger scope is an attempt to lay the foundations of new devices capable of improving the state-of-the-art performances in tracking; in this respect, future developments and further analysis will often be pointed to.

Abstract

Determinare le traiettorie delle particelle è un'operazione che viene effettuata all'interno di vari contesti. Si fa, per esempio, nelle sale sperimentali del CERN dove si studia la fisica delle alte energie, negli ospedali dove si ricostruiscono le immagini del corpo del paziente prima di una terapia, nelle industrie dove si rivelano eventuali difetti di produzione e nei telescopi per studiare i segreti dell'universo. Tra i dispositivi capaci di traccaire particelle, dominano i sensori in silicio, che negli ultimi decenni hanno migliorato le proprie performances sotto ogni aspetto: hanno buone risoluzioni spaziali e temporali, basso consumo, cambiano poco la traiettoria delle particelle passanti, sono relativamente facili ed economici da produrre e sono facilmente adattabili alle diverse applicazioni. In particolare, i recenti dispositivi a pixel monolitici (MAPS), che implementano nello stesso chip sia il sensore che la logica necessaria per l'amplificazione, il trasporto, e l'interpretazione del segnale, si stanno diffondendo, promettendo notevoli miglioramenti.

In questo momento, sia nella comunità scientifica che in quella industriale, ci sono molte realtà che stanno sviluppando dei MAPS. La collaborazione ARCADIA, dell'Istituto Nazionale di Fisica Nucleare (INFN), sta cercando di sviluppare un sensore monolitico che sia orientato ad applicazioni che richiedono grande area e basso consumo. ARCADIA si ispira ad ALPIDE (sviluppato dall'esperimento ALICE del CERN) considerato, ad oggi, un punto di riferimento nel mondo dei MAPS.

Questa tesi introduce inizialmente il processo che ha portato alla creazione del primo sensore di ARCADIA, per poi discutere più in dettaglio l'implementazione del protocollo di lettura, dettato dalle esigenze sperimentali. Inoltre, nell'ottica della validazione del sensore, si presentano i risultati preliminari di una campagna di studio volta all'indagine degli effetti dovuti al danneggiamento del reticolo causato da radiazione non-ionizzante.

La parte centrale di questo lavoro è la discussione di una innovativa architettura di lettura da utilizzare in un MAPS, pensata per applicazioni a potenze estremamente basse (come per esempio le applicazioni spaziali), che permette di ridurre la dimensione dei pixel. L'idea è quella di ricostruire la posizione in cui è passata la particella usando delle funzioni che la proiettino su diversi assi. Questo approccio sfrutta la sparsificazione dei dati prevista dalle applicazioni del sensore. Questa architettura è stata sviluppata dal punto di vista dell'elettronica digitale e le sue performances sono state studiate sia con un approccio matematico (usando un formalismo appositamente creato), sia con un approccio simulativo che riproducesse le condizioni sperimentali. Le simulazioni sono state scritte sia in un linguaggio di descrizione hardware (per la validazione), che all'interno di una struttura puramente software che permettesse uno studio a più alta statistica. Si discutono brevemente anche dei calcoli preliminari sulle performances di questa architettura in termini di potenza. Nel prossimo futuro, si programmano delle prove su silicio.

Questo lavoro è parte di un progetto atto alla validazione del sensore ARCADIA e, inserito in un contesto più ampio, è una prova dedita al miglioramento delle operazioni di tracciamento in silicio.

Contents

Chapter 1

Executive summary

1.1 Overview

In the field of particle tracking, the scientific community routinely uses solid-state pixel sensors to detect and track ionising particles with micrometric spatial resolutions, from elementary particles produced at the CERN **L**arge **H**adron **C**ollider (LHC) [1]–[4], to cosmic rays arriving from distant galaxies detected by space-born telescopes [5]. The very same sensors have also found successful applications in many fields where particle tracking may provide critical information, such as medical imaging [6] or hazardous material security [7].

This thesis focuses on a specific category of pixel sensors, the so-called **M**onolithic **A**ctive **P**ixel **S**ensors (MAPS), which to a first approximation share both their working principle and most of their structure with the pixel sensors commonly found in standard consumer electronics, such as smartphones and cameras, and indeed they are manufactured with the very same microelectronic CMOS imaging processes [8]. The *monolithic* term in the acronym highlights the difference with respect to the *hybrid* pixel sensors, where a pixellated sensor layer, typically made of semiconductor material such as silicon or germanium, is bonded to a matching readout layer, realised with the very same CMOS technology used for MAPS. Hybrid sensors are the mainstay in particle tracking applications, as well as in photon science, where the possibility to fine-tune the sensor layer both in terms of material thickness and material type allows the detection of a wide range of particles, at different energies, and also offers higher radiation resistance with respect to MAPS. However, in recent years, MAPS have reached sufficient maturity to be successfully employed in several scientific experiments, where they offer some key advantages over hybrid solutions, as discussed in this work in detail.

Where scientific MAPS (and hybrid sensors as well) certainly differ from their commercial siblings is in one key design element: they are optimised to read only a very small portion of pixels (less than 1 per mille on average) instead of the entire, full-frame image. Such a procedure, commonly referred as **sparsification** or **zero suppression**, allows to achieve readout speeds otherwise impossible to obtain with traditional full-frame sensors. Sensors

aimed at particle tracking also have, depending on their target application, much more stringent requirements in terms of radiation hardness; i.e., the device capability to survive and operate in hostile radiation environments, such as those found, for example, at particle accelerators, medical beams, or in outer space [9].

The focus of this thesis work *is the study and development of an* ***innovative readout architecture*** *for MAPS aimed at particle tracking, and the characterisation of a sensor prototype device incorporating an innovative, fully-depleted substrate. This work attempts to improve the current state-of-the-art in MAPS in the key areas of power consumption, readout speed, pixel pitch, and verified radiation hardness.*

1.2 Target application and state-of-the-art

In the scientific world, MAPS find prominent applications in the tracker detectors developed for particle physics experiments, and every big experiment at the CERN LHC already uses MAPS [10], or plans to use them in the near future [11]–[13]. These devices are also being used in space-born trackers onboard satellite missions [14]. Tracker detectors usually consist of several stacked layers, arranged in planar or cylindrical symmetry, where each plane registers a passing particle position in space and time; by combining the positions recorded by each layer, it is possible to reconstruct the particle trajectory in 3D space, the so-called **track**. Unlike commercial applications, the reduction of the material budget is of paramount importance to minimise the interference of the sensor with the particle path, as the tracking operation often is only a component of the overall experiment. This, in turn, pushes for the minimisation of the power consumption in order to reduce the thickness of the cooling system [15].

The specifications for tracking MAPS change widely, depending on the application, and there is no way to meet all of them with a single design. Therefore, this work narrows its target to the optimisation of MAPS used in tracking applications that require the lowest possible material budget and, consequently, the lowest achievable power consumption (below $35\,\mathrm{mW\,cm^{-2}}$), together with a small pixel pitch (10 to 20 µm), and moderate radiation resistance (5 kGy, 2×10^{12} 1 MeV $\mathrm{n_{eq}/cm^2}$). This is a reference case for space experiments and some **H**igh **E**nergy **P**hysics (HEP) ones too.

To better frame the *small pixel pitch* figure, it is worth recalling that the current generation pixel sensors installed at the LHC experiments have a pixel pitch ranging from 150 µm [16], [17] down to 28 µm [10], while pixels found in commercial imaging devices usually are in the 1 µm to 10 µm range. The above "small" pitch attribute therefore refers to the pixels sensors typically employed in high energy physics tracking applications, where pixel dimensions below 20 µm are considered small. The case for a finer pitch is prompted by the need of improving the spatial resolution of the particle origin position to the 1 µm level, which implies a pixel pitch on the order of 10 µm size, when utilising at least 3 measurement points in space, each characterised by a $\frac{10}{\sqrt{12}}$ µm uncertainty (Particle track reconstruction resolution approximately goes as $\frac{\sigma}{\sqrt{N+1}}$ for an ideal detector, where σ is the single layer

spatial resolution and N is the number of tracking layers, as a consequence of a simple linear interpolation).

From the point of view of radiation damage, MAPS are considered moderately radiation resistant when compared to tracking detectors at accelerator facilities: the benchmark for high radiation levels is established by high repetition rate (40 MHz) proton-proton collision, where over the lifetime of an experiment (typically 5-10 years) the integrated radiation levels may reach 10 Mrad and 2×10^{16} 1 MeV n_{eq}/cm^2respectively. Conversely, experiments where the interaction rate is lower [10] and/or the colliding particles are leptons instead of hadrons [18], typical integrated radiation levels over the experiment lifetime are on the order of 500 krad and 2×10^{12} 1 MeV n_{eq}/cm^2.

For what concerns the power consumption, the current benchmark for scientific MAPS is the ALPIDE sensor developed by the ALICE collaboration [10], which has a reference power consumption of 35 mW cm^{-2} (depending on various operational modalities, this value may change up to a factor 2 in both directions). The hybrid pixel sensors planned for installation in the near future at the ATLAS and CMS experiments [19] [20], has much larger consumption, in the order of 1000 mW cm^{-2}. Such larger power dissipation is due in part to the different technology (hybrid versus monolithic) and in part to the different specifications, the maximum hit rate in particular, which greatly exceed that of the ALPIDE sensor. These considerations about the power dissipation highlight how, aiming for minimum consumption and a small pixel pitch, will require a trade-off in terms of maximum particle rate. For reference, the aforementioned ATLAS and CMS sensors have been designed to handle a maximum rate of 3000 MHz cm^{-2}, while the ALPIDE can withstand up to 10 MHz cm^{-2}, more than a two orders of magnitude difference.

This work aims at improving the experiments in which ALPIDE is currently used; hence, the reference specifications given above are an improvement over the current gold standard in the field. Such a device would meet the requirements of a few different applications, for example:

- **precision experiments at next generation lepton colliders**: The interaction between leptons, such as electrons or muons, is much cleaner and lepton colliders produce a less dense particle field with respect to hadron machines (like the LHC). To best record these fine details requires sensors with the lowest possible material budget, and hence power dissipation. The power level goal is the one which would allow air cooling; i.e., below 20 mW cm^{-2} [21] (or even in-vacuum radiation only cooling - see space applications). The sparser particle field allows relaxing the radiation resistance of the exposed device with respect to the present detectors installed at the LHC [22] by 2 orders of magnitude, rendering MAPS a viable choice;
- **medical tracking**: The usage of light ions (protons, He, Ca) medical pencil-beam is the most advanced and effective radiotherapy available for cancer treatment. Protons of the order of 250 MeV will be used as well to render 3D images of the body, like in a CT scan, with the added benefit of a much lower dose to the patient, and a better tissue density resolution [23] (see Section 2.4.4). Such 3D imaging requires precise proton tracking, just like in a HEP tracking particle experiment, with the caveat that

in order to avoid excessive multiple scattering of the protons requires that the sensors be as thin as possible;

- **space applications**: Space telescopes tracking cosmic radiation embed at their heart a particle tracker, conceptually identical to those found in ground-based particle physics experiments. Like any space application, energy consumption is paramount, due to the limited space vessel power sources as well as problematic dissipation issues. For this reason, all major space telescopes still use micro-strips because of power budget constrains [24], the current limit being about $5\,\mathrm{mW\,cm^{-2}}$, far below the current state-of-the-art ALPIDE power budget;
- **industrial applications**: Industry is interested in imaging, metrology, tomographic reconstruction, and range-finding applications. For this purposes almost exclusively visible, infra-red, and x-ray photons are used because of the complexity, cost, and restrictions for using other radiation sources that are employed in scientific, medical, and military facilities. Whereas silicon has efficient photon detection in the visible and near-IR band, it is almost transparent to higher energy photons: in scientific applications the sensors of choice for x-ray imaging are the hybrid pixel sensors, with Germanium or other high-density semiconductor materials used for the sensor layer. However, the intrinsic high costs of hybrid pixel productions renders this solution unpractical for industrial applications, where instead x-ray panels (based on TFT monitor technology) are the most widespread sensor, with the trade-off of low resolution (1 mm) and speed (10 Hz). MAPS, although in principle not optimal for x-ray detection, may be adapted to improve x-ray detection efficiency (for example, by reaching full depletion [25]); in addition their low production cost (compared with other technologies) boosts their convenience when there is the necessity to instrument large detection area, as in industrial Computed Tomography systems.

The challenge in meeting these specifications scales with the device area: whereas a very small sensor of few square millimetres may be (almost) promptly designed, realising a single-piece detector of several square centimetres with the target performance level requires pushing the state-of-the-art. It is worth stressing here again that, in many applications like those just presented, tiling several small sensors is not an option, usually due to extremely tight material budget constraints and, therefore, the goal is indeed maximising the area covered by a single sensor. Likewise, the maximum sustainable rate affects also the design complexity: at extremely low rates, large area sensors with extremely low power dissipation are indeed possible, as exemplified by astronomical CCDs [26].

Looking at the present state-of-the-art, and reviewing the requirement for some planned experiments in the field of high energy physics [27] [28], it is apparent how a single sensor having an active area equal or bigger than $10\,\mathrm{cm^2}$, and capable to withstand a maximum rate of $20\,\mathrm{MHz\,cm^{-2}}$ or more, would be able to meet the requirements for such future scientific applications when coupled with small pixel pitch and low power consumption.

Summarising, the target specifications for the device envisioned by thesis are the following:

- pixel pitch: $O(10\,\mu\mathrm{m})$

- power consumption: $\leq 35\,\mathrm{mW\,cm^{-2}}$
- radiation tolerance: $\geq 500\,\mathrm{krad}$ and 2×10^{12} 1 MeV $\mathrm{n_{eq}/cm^2}$
- active area: $\geq 10\,\mathrm{cm^2}$
- maximum rate: $\geq 20\,\mathrm{MHz\,cm^{-2}}$

1.3 Methodology

The idea pursued in this work in order to meet these goals heavily relies on performing sparsification at the architectural level: in the proposed architecture, called **hash**, a hit pixel asynchronously triggers multiple signals through a net of multiple metal wires, where each line is shared among different subsets of the pixel array. These wires take the signals to the periphery, where they produce a so-called **hash code**, which is used to reconstruct the hit pixel address. This approach reduces the number of in-pixel transistors necessary to communicate the hit position to the periphery, when compared to the current state-of-the-art architectures, leading to the following improvements:

- fewer transistors occupy less space; hence, make the pitch shrinking feasible;
- fewer transistors reduce the power consumption, as there will be both less leakage and less transistions on average;
- fewer transistors free more layout area for routing, simplifying the realization of a scalable architecture for very large area sensors.

The trade-off of such solution is the maximum sustainable particle flux: according to the simulations discussed in this work, such limit is however higher than the current benchmark set by the ALPIDE sensor collaboration.

The proposed hash architecture is indeed an abstract concept, not necessarily limited to silicon pixel sensors, that might be successfully applied for the readout of any matrix of binary values where sparsification conditions are met. In this work it is applied to MAPS, and several implementations flavours are discussed.

1.4 Results

The **hash readout architecture** has been completely developed at the **R**egister **T**ransfer **L**evel (RTL). The design has been validated through both mathematical analysis and intensive Monte Carlo simulations, and shows no major concerns; the pixel area was estimated post-synthesis (LFoundry 110 nm technological node); the power consumption has been evaluated through analytical calculations performed manually (communication power) or using proper circuit synthesis and simulation tools (switching and leakage power) for the pixels. The maximum sustainable flux was estimated through a software fully replicating the architecture in a computationally-faster environment.

The results obtained clearly depend on the implementation chosen (flavour), as several

have been outlined to try and assess the potentialities and the limits of this innovative approach. For the most conservative case that was studied in detail, the results are:

- a full sensor sector sized $0.8 \times 12.8\text{mm}^2$ tilable on the long side. Currently, ALPIDE sensor has a size of $15 \times 30\text{mm}^2$; hence, a first MAPS implementing of the hash readout could have a similar size by tiling 32 such sectors (or a square one by tiling 16 sectors). The design is scalable and can be adapetd to bigger devices;
- a complete pixel digital part sized $\approx 125\,\mu\text{m}^2$ per pixel, compared with the $\approx 250\,\mu\text{m}^2$ of the ALPIDE sensor pixel digital part;
- an estimated power consumption for the sensor pixels bulk $\leq 10\,\text{mW}\,\text{cm}^{-2}$, to be compared to $35\,\text{mW}\,\text{cm}^{-2}$, the total power consumption of ALPIDE;
- an efficiency curve that starts dropping at about 1×10^8 particles per square centimeter, assuming that a cluster is in one pixel region; a tracker in ALICE like the ALPIDE has an expected peak flux of 2×10^6 particles per square centimeter [29] (even in a very pessimistic corner where the hit is 4-regions large, the efficiency remains high in ALPIDE working context).

These figures can be further improved by implementing some extra design optimizations, which are discussed at the end of this work, and which give confidence on how this architecture might effectively improve the current state-of-the-art. Such bolder solutions would make it feasible to:

- create bigger sensors trying more complex place-and-route;
- reduce the power consumption changing the readout-and-reset scheme taking further advantage of the hash approach;
- shrink the pixel further by optimising the place-and-route;
- push the rate limit further by optimising the periphery.

From the radiation-resistance point of view, the performances of the reference LFoundry 110 nm modified CIS technological node [30] has been measured by using a fully depleted MAPS (ARCADIA) that is developed in the very same technology planned for the hash readout architecture. Some tests discussed in this work showed how from the bulk damage point of view the sensor is still functional at around 15 °C for an irradiation level of 1×10^{13} 1 MeV $\text{n}_{\text{eq}}/\text{cm}^2$. The damage due to ionising radiation has not been quantified in this thesis but an experimental campaign [31] measured a two-fold increase of the noise (still usable sensor) after a dose of 10 Mrad.

1.5 Outlook

Chapter 2 introduces the scientific interest for the issues that are addressed in this thesis. Section 2.1 presents the history of trackers as well as the driving principles that make charged particle tracking possible; Section 2.2 presents the three ways that are used to move the information around the array (CCDs, hybrid solutions, MAPS); Section 2.3 lists the three problems that need to be addressed in order to develop a working silicon tracker (sensor design, analogue design, digital design) together with few necessary thoughts on the choice

of the technological node; Section 2.4 shows four experiments and points out why silicon tracking is important for them and how it is implemented.

Chapter 3 presents the commissioning work of the ARCADIA sensor, a fully depleted MAPS, in particular in its first version ARCADIA-MD1. This work is a concrete example of the steps necessary to test such devices. Section 3.1 gives an overview of the ARCADIA project, showing why it is interesting for the tracking community and briefly presenting the work done to implement it; Section 3.2 presents the development of the data acquisition system of ARCADIA, together with some experimental plots; Section 3.3 presents the results of a neutron irradiation campaign to test the resistance to bulk damage of the 110 nm modified CIS [30] technology node.

Chapter 4 finally introduces the hash architecture, framing it among other digital readout architectures. Section 4.1.1 shows how to rate CMOS circuits in terms of power consumption and time delays, focusing in particular on the binary readout architectures that are the subject of this thesis; Section 4.2 presents three architectures: an abstract clockless column readout architecture, the ALPIDE readout architecture, and the hash architecture, rating them using simple algebraic parameters; Section 4.3 presents a mathematical framework created to rate the several connections (called projections) depending on how well they perform for hash purposes.

Chapter 5 discusses the implementation of the hash architecture and estimates its performance. Section 5.1 presents the physical implementation developed for this work; Section 5.2 shows the analytical calculations performed studying the technological libraries to estimate both the power and the timing performance of the device; Section 5.3 presents the set of Monte Carlo simulations performed to assess the correct functioning of the architecture and the maximum particle flux; Section 5.4 presents some alternative hash architectures that may further improve the performances.

Appendix A gives a theoretical presentation of the issue of radiation damaging for silicon sensors; Appendix B shows the smart coincidences seeker routine used to spot signals in more than one ARCADIA sensor; Apprendix C shows an example of a single hit propagating through the Monte Carlo software developed to evaluate the performance of the hash readout architecture.

Chapter 2

Charged particle tracking

Whenever a charged particle traverses matter, it deposits energy by interacting with it. Particle trackers are devices capable of intercepting these various energy depositions in order to reconstruct the particle trajectories. Generally, tracking refers to a non-invasive operation: the purpose is quite often to understand where the particle is going without disturbing the trajectory itself, as tracking is in many cases only a component of a more complex measurement. This is not completely achievable, as the particle must interact with matter in order to be spot, and such an interaction leads to changes in its kinematics, but this effect can be minimized. In the last century, the tracking of charged particles has been one of the most demanding challenges in the field of particle physics. In this introductory chapter, first an overview on the tracking mechanisms and a historical roundup is given in Section 2.1; next sections focus with more details on tracking by using silicon pixel sensors, the subject of this thesis. In Section 2.2 the common readout paradigms used for tracking is silicon pixels are listed, and the development approach generally followed is described in Section 2.3. Finally, Section 2.4 lists some experiments where silicon trackers are used and review what their characteristics are.

2.1 Tracking principles and history

The physics of charged particles tracking

The equation that describes the energy loss of a charged particle in matter is the Bethe-Bloch equation, known since the 1930s [32]. Bethe-Bloch is a formula describing what is referred to as **ionising energy loss**; i.e., the energy that is continuously lost by the charged particle as it progresses through the material due to its interactions with the electrons in the material. These electrons, which receive energy, are excited and easily detached from the atom with which they are bound. This leads to the production of a free electron and a quasi-particle called hole; i.e., a missing electron in matter. In most cases, a free electron jumps back into the hole right after its creation in a process called **recombination**, in which

case there is no evidence of the passage of the particle. But in some cases the effect of the particle is persistent, as some freed electrons do not immediately recombine; in this situation the electron-hole pairs may ultimately be detected and used for particle tracking. Ionising energy loss is not the only process that can lead to energy loss in matter for a charged particle; nevertheless, it is the most probable and easy to use.

Neutral particles like photons and neutrons can be detected as well, but this can happen only if these particles create a charged counterpart that interacts with matter. Photons interact via the photoelectric effect, Compton scattering, or pair production; in all these three processes, charge is generated inside the material (positrons and/or electrons); however, the original photon is destroyed. Neutrons interact via strong nuclear interactions and can create ionising nuclear recoils, fragments, and other neutrons. Here too, the original neutron is often destroyed, or its kinematics deeply modified. The focus of this thesis is the study of trackers for charged particles; nevertheless, they can be successfully employed to also detect these neutral particles.

The first particle trackers

The first devices that could track single particles were used to detect radiation in the natural environment, natural radioactivity, and especially cosmic rays. Trackers development started from nuclear emulsions first developed around 1910 [33], [34], and evolved to bubble chambers (1952 [35]), spark chambers (1954 [36]), multi-wire proportional chambers (1968 [37]), drift chambers (1971 [38]), and time projection chambers (1974 [39]).

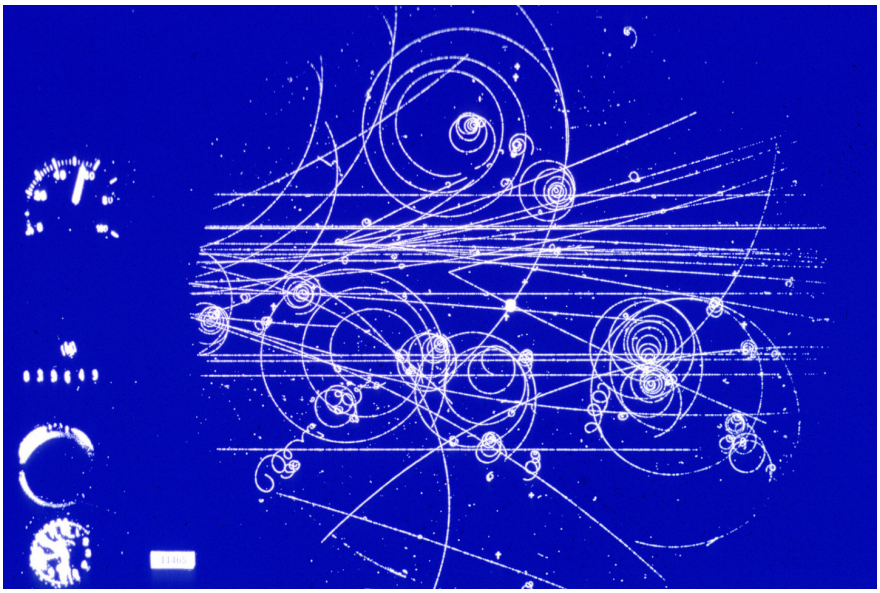


Figure 2.1: A 1960 image from the CERN liquid hydrogen bubble chamber of a lambda decay. Taken from [40].

As a reference, the working principle of bubble chambers is presented (see Figure 2.1): the tracker is a device containing an unstable liquid heated above its evaporation temperature.

If the liquid container is clean enough, there are no suitable places for the bubble enucleation needed for the liquid to begin the transition to the gas phase. However the energy deposition by ionising radiation leads to the generation of free electrons and holes that create regions where the liquid begins to locally boil; this gives rise to the formation of longer-lasting visible bubbles along the particle trajectory. By taking photos from different directions of the bubble chamber right after the passage of the ionising particle (few milliseconds later), it is possible to actually track the charged particle in three dimensions.

Tracking in silicon

A big boost to tracking techniques arrived thanks to the modern photolithography processes developed by the semiconductor industry, which led to the development of the first silicon trackers in the 1980s. These devices rapidly became the focus of the tracking communities in various fields, such as **H**igh **E**nergy **P**hysics (HEP), astro-particle physics, nuclear physics, and medical physics. Silicon is by far the most widely used semiconductor today because of many features: it is very abundant, relatively easy and inexpensive to manufacture, stable at room temperature, and its oxide is a really good insulating material. Historically [41], the very first silicon detectors were used in Berkeley [41], Dubna [42], and CERN [43], [44]. The complete treatment of the working principles of silicon as a charged particle detector can be found in several academic textbooks that thoroughly discuss this matter [45], [46]. Silicon is a semiconductor: i.e., it has a finite gap (about 1.1 eV) between the valence band (a band full of electrons when the device is at absolute zero temperature) and the conduction band (where the electrons can move through the crystal). By adding some dopants, new possible states can be placed within the gap, and the silicon behaviour can be engineered. There are two different types of dopants used: a *p-type* silicon contains some atoms of an element of the third chemical group (such as boron or gallium), while a *n-type* silicon contains some atoms of an element of the fifth chemical group (such as arsenic or phosphorus).

When a p-type silicon is joined to an n-type silicon, a so-called pn-junction is formed; there is a step in the electron density curve, hence charges (electrons and holes) start diffusing through the junction until there is dynamic equilibrium between the diffusion pressure and the electrical pressure due to the charge accumulation. Around the junction, a **depleted region** is created: a portion of silicon where all mobile charges have been removed; i.e. there are no free charges (see Figure 2.2) but only naked ionised dopants. The application of an external voltage called *reverse bias* can further extend this depleted region. When the ionising particles travel through such a depleted region, the generated charges do not recombine as both the electrons and the holes follow the local electric field and are promptly pushed away from each other. Thanks to this effect, a pn-junction (also known as a diode) can be successfully used as the basis for a tracker of charged particles.

The charges generated inside a depleted region are collected and integrated on a proper capacitance that can be read out; the particle passage is pinned down and can be measured. The equation defining the voltage produced by charges in a capacitor is the following:

$$V = \frac{Q}{C} \tag{2.1}$$

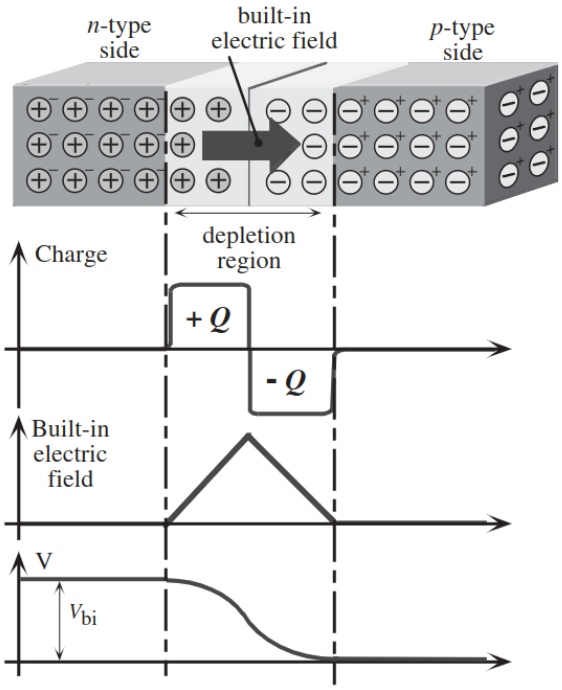


Figure 2.2: A scheme representing a pn-junction. Plots for charge, electric field and potential distributions are shown as well. Taken from [46].

This equation shows how, for a fixed number of charges generated by the impinging particle, the smaller the capacitance, the higher the voltage to be read out; thus, the signal read will be higher and the device will be better-performing[1]. A smaller capacitance can be achieved by reducing the size of the electrode that collects the charges; exactly for this reason the first silicon detectors were **microstrips**: long diodes (strips) are created along a direction and then charges are read through a long electrode (see Figure 2.3). Because of this segmentation, the capacitance is lower compared to a single, large device, and the signals can be easily retrieved by accessing the electrodes at one of their ends. The first microstrip devices were not used as trackers according to modern interpretation, but were instead active targets: the impinging particle reacted with the silicon, and the signal was read out on the electrode that was closest to the interaction point. The first use of silicon for tracking dates to 1981, with the experiments NA11/NA32 [47], where six planes of silicon microstrip detectors were mounted to measure the vertex position.

The use of a tracker that generates an electric signal requires a **readout** system, an aspect that is discussed throughout the main chapters of this thesis (Chapter 4 and Chapter 5). To read out a microstrip sensor, it is necessary to access the charges stored on the long capacitors. It is possible to connect individually to each one of them via a wire bonding and use a separate device to perform the necessary operations to make this voltage signal readable; i.e., the amplification, shaping, and digitisation of the signal.

[1] This is an over-simplification. A lower capacitance improves both the signal-to-noise ratio and the timing resolution, while a higher capacitance improves the charge collection efficiency and the spatial resolution.

Microstrips provide several advantages over the other tracking techniques listed before, in particular:

- fast response (of the order of the ns);
- high segmentation, hence good spatial resolution (of the order of tens of µm);
- good linearity and energy resolution (at room temperature of the order of the keV);
- simplicity in operation.

For these reasons, microstrips have been widely used since their invention and are still commonly found today.

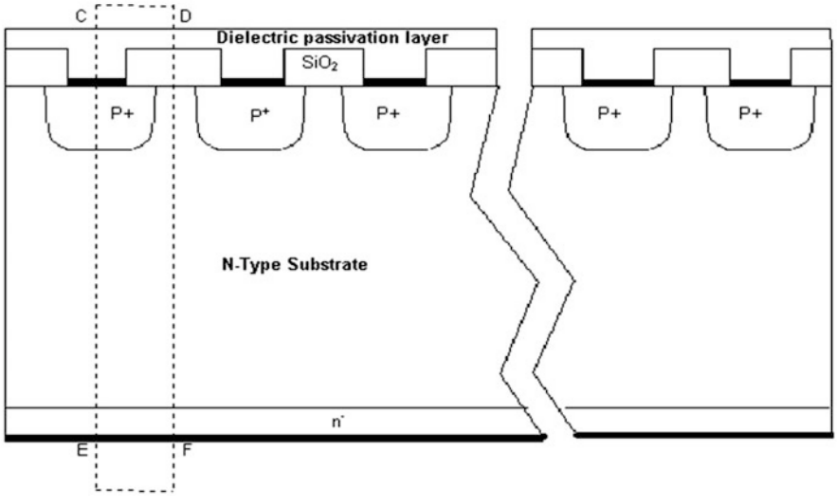
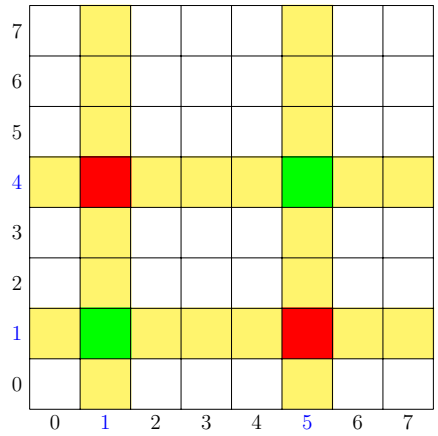


Figure 2.3: The cross-section of a silicon microstrip detector. Single long diodes are created in silicon, each one with its own electrode. The strips, in this image, are oriented as perpendicular to the page. By reading the voltage on the electrode it is possible to measure the energy deposited in the sensor. Taken from [48].

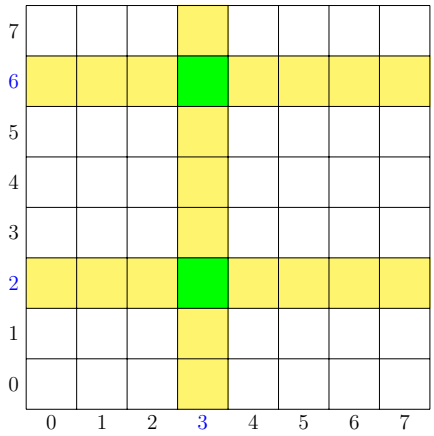
Silicon pixel trackers

One of the main limitations of the microstrip approach lies in the fact that they can measure only one direction of the particle position; a second one can be inferred by the tracking plane position, considering that the sensor is usually very thin $\mathcal{O}(10 - 100\,\mu\text{m})$; the third coordinate cannot be accessed. For this reason, pairs of microstrip detectors are often used: one sensor is placed to have the strips aligned along the x-axis, and shortly after (often in the same silicon die), a second sensor has the strips aligned along the y-axis. By reading both sensors, it is possible to obtain the three-dimensional coordinates of the point where the ionising particle passed. This approach fails for cases with many particle tracks (**high multiplicities**), that are called in this thesis **data collisions**. If there are two charged particles crossing the detector almost at the same time, two data concerning the x position and two data concerning the y position would be obtained, and it would not be possible to pair these data obtaining the two 3D points: there will be both two correct reconstructions

and two wrong ones (in most cases, see Figure 2.4). This issue can be tackled by changing the approach: instead of having long electrodes, it is possible to construct small square-like electrodes, and read both the x and the y at the same time. These devices are called **pixel sensors**. Formally, the definition of a silicon pixel sensor used in this work is: "a silicon tracking sensor using a pn-junction to detect ionising particles and divided into small rectangular or square regions, usually with a size of the order of tens of µm".



(a) A problem when 2 hits arrive on a matrix.



(b) Not a problem when 2 hits arrive on a matrix.

Figure 2.4: Two possible situations where there are two hits in a matrix when there are two projections: one along x and one along y. In Figure 2.4a four addresses are reconstructed, but whereas two of them are right (green ones) two are wrong (red ones). In Figure 2.4b only right addresses are reconstructed. In yellow, the pixels that would generate one of the signals actually seen.

The main issue when using pixel sensors, in comparison with microstrips, lies in their readout: It is relatively easy to access to charges on a long electrode by connecting to one of its ends, but in pixel sensors there are far more electrodes, and they are smaller and distributed throughout the array, hence difficult to reach. There are three approaches pursued to readout a pixel sensor, as is discussed in the next sections: **C**harge-**C**oupled **D**evices (CCDs), hybrid devices, and **M**onolithic **A**ctive **P**ixel **S**ensors (MAPS). Historically, in 1982, a Rutherford Laboratory Group showed that a CCD can be used to successfully measure the position of minimum ionising particles [41]: the first pixel silicon tracker.

Silicon pixel performances

When comparing the performances of different silicon pixel sensors, the following parameters are important to take into account:

- **Efficiency**: an ideal tracker can detect any particle that crosses it; a physical tracker will miss some of them. Efficiency depends both on the sensitive fraction of the sensor and on the noise level: the higher the noise, the harder it is to recognise the true signal generated by an impinging particle;

- **Minimum energy**: the more energy is deposited in the silicon lattice, the more likely it is for the sensor to actually detect the particle passage. The ideal tracker in most HEP experiments is meant to detect **M**inimum **I**onising **P**articles (MIPs): charged particles with a high enough energies to deposit the minimum quantity according to the Bethe-Bloch equation[2]. This minimum energy threshold is usually dependent on electronic noise of the readout chain and the pixel capacitance;
- **Energy deposit resolution**: some pixel trackers also provide information regarding the energy deposited by the ionising particle along its passage through the sensor. This information has its own resolution, depending on many factors, such as the linearity of the sensor collection capacity (as a reference number, ATLASPix has an energy resolution of 7% at 30 keV [51]);
- **Spatial resolution**: the main purpose of the tracker is to measure the position of the impinging particle. In the case of silicon pixels, this information is bi-dimensional (and the third dimension can be inferred by the sensor positioning). The precision over the axes defines the spatial resolution. This number must match the one required by the experiment[3]. The main contribution to the spatial resolution of a pixel tracker is the size of the pixels, also called **pitch**. A good pixel sensor for charged particles can reach a spatial resolution of about 1.5 µm [52];
- **Timing resolution**: whenever a signal is generated in the sensor, a time label is also associated with the spatial variable. Depending on various conditions, such as the signal shaping circuitry and the free charges collection speed, the timing resolution varies. The most precise silicon pixels now approach what is called 4-dimensional tracking, i.e., a really good precision also on the timing dimension, reaching resolutions of the order of 30 ps [53];
- **Power consumption**: the sensor functioning consumes energy. The lower the power consumption, the better it is for all the applications. Some experiments have very strict constraints: this is the case of space, where the available power is limited and the heat dissipation in vacuum is very challenging, and the case of HEP, where the power consumption translates into potentially thick dissipation systems. ALPIDE is one of the best performing sensors from this point of view, and has a power consumption around 35 $\mathrm{mW\,cm^{-2}}$ [54];
- **Rate capability**: a pixel sensor is able to correctly identify and distinguish particles up to a certain flux, beyond which the pileup can lead to incorrect or incomplete hits readout. The maximum rate a device is expected to withstand is important in choosing a suitable detection technique. One of the highest-whithstanding silicon pixel tracker is CMS inner tracker, able to disentangle up to 6×10^8 particles per second per square centimeter [55];
- **Material budget**: in several applications, tracking is but a part of the overall measurement to perform on the particle; hence, the tracker must be nondestructive. Nev-

[2]For silicon, a MIP deposits 388 $\mathrm{eV\,\mu m^{-1}}$ [49], hence it creates on average about 108 electrons per µm but, most of the times, a value closer to 75 per µm, as the distribution is Landau-shaped [50].

[3]In some applications, e.g., spectroscopy, information on the particle energy is retrieved through a position measurement, therefore in these cases the spatial resolution of the detector strongly affects the energy resolution of the experiment.

ertheless, the passage of the impinging particle inside the tracking system inevitably perturbs and degrades the particle kinematics, as the physical process of detection requires some energy deposition in silicon. As a rule of thumb, the thinner the tracking system, the less the kinematic degradation (as a reference, ALPIDE innermost layers have a material budget of 0.3%, meaning that the detector depth equivalent thickness is 0.3% of the radiation length X_0);

- **Radiation hardness**: the detector is subject to radiation during its operation; over time, the radiation to be detected also damages the sensor, worsening its performance. Damage to the device can be measured in terms of total ionisation dose effects and bulk damage effects. Some devices are more radiation resistant than others. As a reference, for HL-LHC [48] CMS tracker is expecting a **N**on-**I**onising **E**nergy **L**oss (NIEL) up to 2×10^{16} 1 MeV n_{eq}/cm^2[56]; the collaboration is already addressing such an issue studying radiation-resistant possibilites [57].

In actual silicon pixel sensor implementations, there is a trade-off between these characteristics. The numbers reported here as references show some of the best performances of silicon pixel sensors for a given quantity, but many refer to different sensors. This is because it is highly unlikely for a sensor developed for extremely good timing resolution to be also low power-consuming, or to have a great radiation hardness; and similar correlations exist between all the quantities introduced. Sensor development must be fine-tuned with the target application.

2.2 Pixel sensors types

To address the issue of accessing the electrodes on the pixels, various approaches have been developed. In this section, the three most widely-used methods are introduced: CCDs, hybrid devices, and MAPS.

Before discussing how to move the information on the array, it is necessary to clarify what the information actually is. There are two different types of silicon pixel trackers: **analogue** and **digital** ones. In the analogue case, the information is the height of a voltage signal proportional to the energy deposited in the sensitive region of the silicon; in the digital case, on the other hand, this height is compared to a set of thresholds, and a digital word represents the highest threshold that is lower than the signal itself. This work focuses on digital pixel trackers, in particular with a **binary readout**: only one threshold is present for every pixel; therefore, the information is just whether an ionsing particle passed through a given pixel or not; this is called **binary hit/no-hit fashion**.

CCD

CCD stands for **C**harge-**C**oupled **D**evice and represents a way to move data along a physical space. CCDs were invented in the 1970s [58] and were actually developed for memories, but later their most famous application was found in imaging (and tracking).

Once the charge is collected in the pixel, it is momentarily saved in a built-in capacitance and stored only when created within a specific time window, named the integration time. When the device is not sensible anymore, a voltage difference is produced row by row to start the flowing of electrons toward the higher voltage. If the voltage is set correctly, the electrons move from one pixel to the one below until they reach a readout node at the end of the array, where this charge signal is converted to a voltage signal and digitised. A scheme of the CCD working principle is reported in Figure 2.5: electrons are slowly moved toward the bottom row where a high speed charge transfer brings them to the bottom right of the device, where their voltage signal is finally read. For CCDs, as in microstrips, the front end is external. The signal read from the single pixel is a charge signal, and is later processed properly before the connection to the computer.

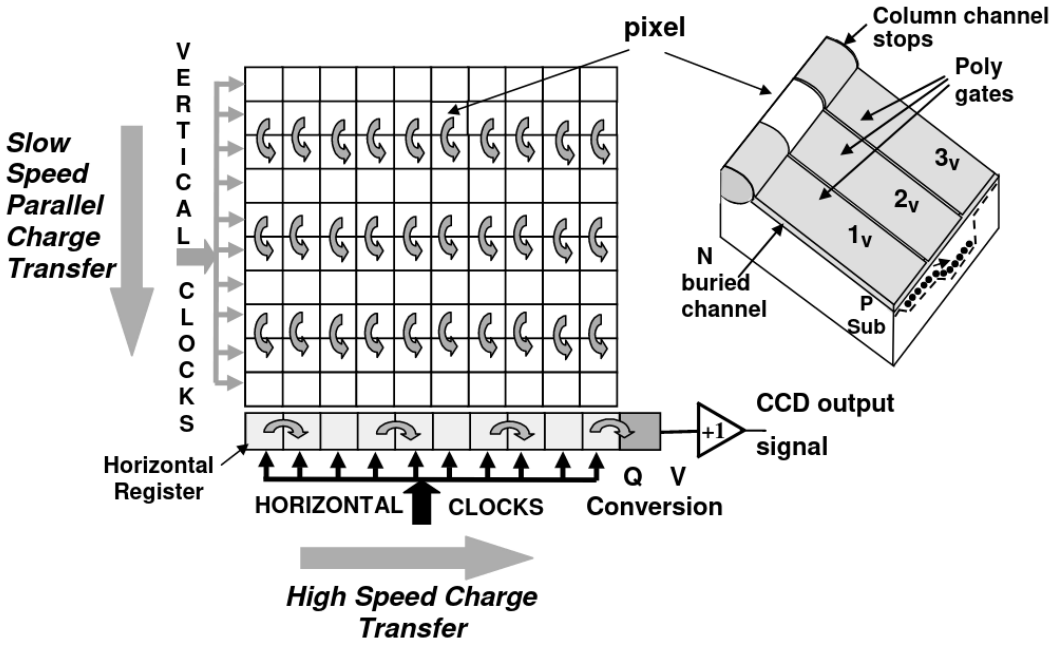


Figure 2.5: A schematic of CCD readout principle. In this scheme, information is transported vertically up to the bottom lane, where is transported from left to right to the readout pixel. Taken from [59].

Hybrid devices

An alternative approach to moving the charge towards the edge of the array is to read each pixel directly, similarly to what is done for microstrip sensors. This pushes the need for a structure, separated from the sensor itself, that connects to it pixel-by-pixel.

This is exactly the approach of hybrid devices: an ASIC (**A**pplication-**S**pecific **I**ntegrated **C**ircuit) is developed with the same size of the device, the same pitch, and the proper pads to connect to the sensor. Then, this ASIC is aligned with the pixel sensor, and metal-bump bonds are created to connect independently every pixel of the device. This approach was

made possible by several advances in the field of assembly; the most recent technologies allow microbumps as wide as 2.5 µm that are only 5 µm apart [60]. In Figure 2.6 a representation of a hybrid pixel detector is shown.

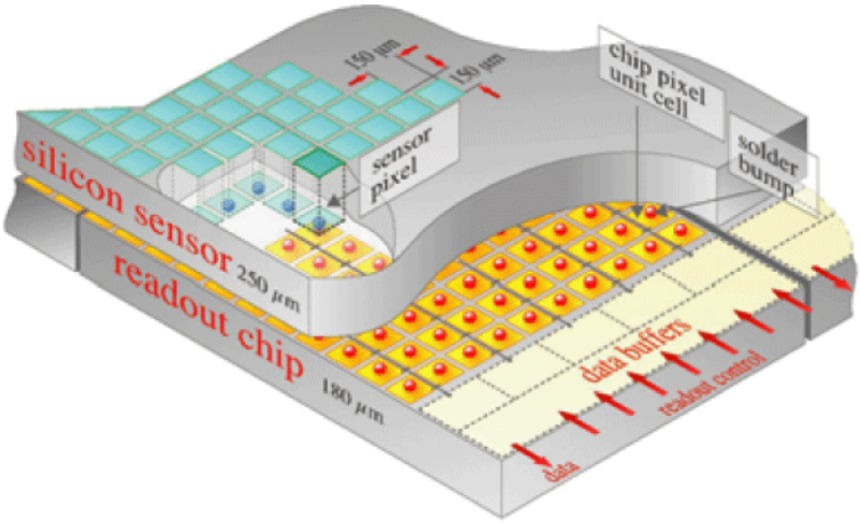


Figure 2.6: A schematic of a hybrid pixel detector. The silicon section is bonded with the readout chip (at least) once for every pixel. Taken from [55].

MAPS

Another possibility to create a silicon pixel tracker is to take advantage of the peculiar properties of silicon as a material. Silicon implanted with pn-junctions, good for the detection of charged particles when depleted, can also host transistors capable of creating both analogue and digital circuits; i.e. both detectors and nMOS/pMOS structures are created on the same die of silicon. This is commonly referred to as CMOS (**C**omplementary **M**etal-**O**xide-**S**ilicon) imaging technology. More details on CMOS circuits are given in Section 4.1.1.

This technology is capitalised to create MAPS: **M**onolithic **A**ctive **P**ixel **S**ensors (also called CMOS sensors): there is no external ASIC as in hybrid implementations, but the very same silicon device that performs the detection of the particle also implements the logic necessary to amplify, transport, and interpret the charge signal correctly. A MAPS actually works as a standalone object: when it is operated correctly, it performs all the operations needed for tracking from the charge creation up to the digitisation of the signal. An example of a MAPS cross section can be seen in Figure 2.7, where transistors can be seen on top of the sensitive region. These transistors are used to implement all the operations needed for the device to function. MAPS is an example of **readout integration**: the readout does not require an external device, but the signal is already read from the sensor as pre-processed and ready for analysis.

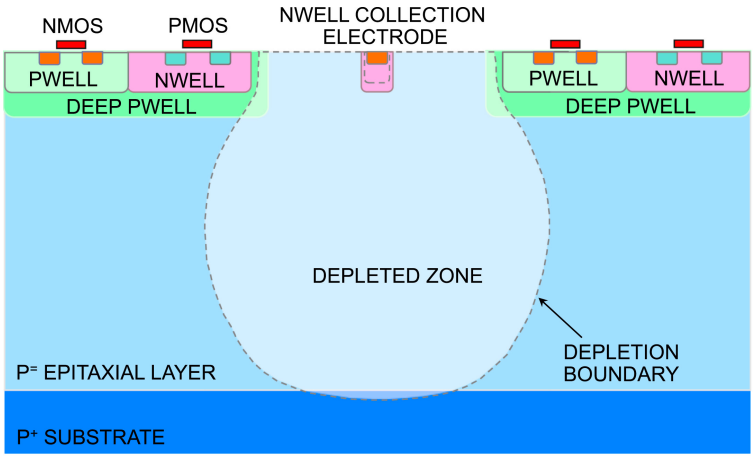


Figure 2.7: Cross section of TowerJazz 0.18 µm CMOS process for imaging. Taken from [61].

Pros and cons

Over the years, all three different flavours have been used, but as of yet there is no one technique that clearly prevails over the others. The CCD approach dominated the market until ∼20 years ago, and is currently used mainly for X-rays tracking; at the moment of writing, most nuclear and HEP applications use hybrid detectors or MAPS, while many astroparticle experiments continue to use CCDs.

The main differences are the following:

- in a CCD all pixels are usually read, while with hybrid devices or MAPS it is possible to take advantage the **sparsification** of data: the fact that a passing particle usually creates signals on a relatively small number of pixels, much lower than the total number of pixels of the device; MAPS and hybrids enable optimised transmission of data;
- CCD and hybrid quantum efficiency is usually better, as the CMOS technique needs to occupy part of the sensitive area with the transistors;
- Hybrid devices have the worst material budget as the material of the sensor plane is then followed by the additional material of the readout ASIC. In the other two cases, the impinging particle only passes through the one sensor plane;
- MAPS and hybrid devices have the lowest power consumption, as only useful information is transported to the outside of the sensor and there is no need to swap often voltages like in CCDs;
- Hybrid devices have the worst pixel pitch: whereas it is possible to exploit industrial processes to create very small pixels into CCDs and MAPS, it is quite hard to create such small pixels in hybrid devices, due to the technological limits of the bonding process (some very recent developments implemented bonds with a pitch as small as 3 µm, partly solving these limitations for hybrid solutions [62]);
- MAPS are considered the cheapest options, as their production is cheaper than CCDs [63]–[65] and, compared to hybrid solutions, there is no need to develop an ASIC and then bond it;

- CCDs are usually slower than MAPS or hybrid devices, as the information must travel queued through the whole sensor.
- Hybrid design allows for a great level of versatility, as the same readout ASIC can be used in multiple experiments simply swapping the sensor plane; on the other hand, they usually suffer from a higher electrical noise, due to the parasitic capacitance introduced by the bump bonds connecting the sensor to the electronics.
- Hybrid designs usually show good radiation tolerance, MAPS show intermediate behaviour, and CCDs are very sensitive to radiation degradation. This is mostly due to to the fact that the architecture in a CMOS circuit (implemented either in the sensor itself in a MAPS or in an external ASIC) is designed to be radiation-resistant, while in CCDs this degree of freedom is not possible.

Looking at the consumer applications in Figure 2.8, dominated by the sensors in the visible range used for cameras, almost all recent imaging applications employ CMOS, mainly for power and price reasons.

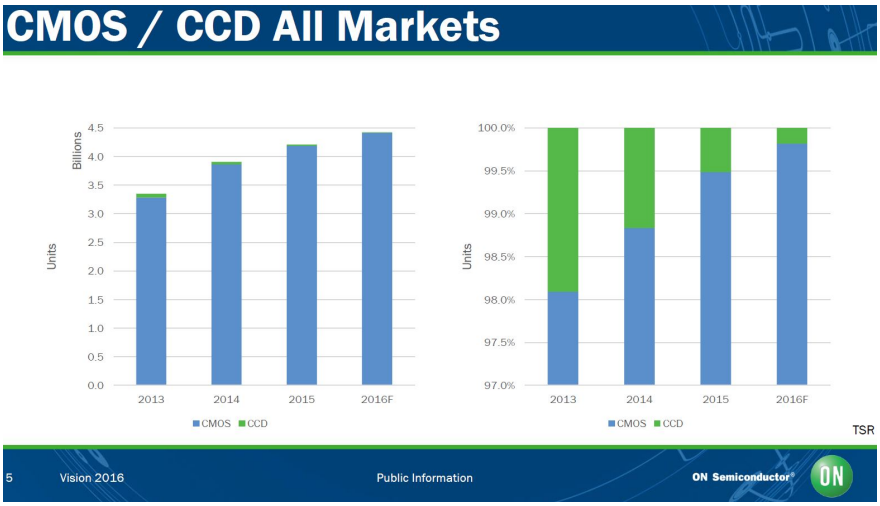


Figure 2.8: Comparison between CCD and CMOS (MAPS) market in industry few years ago. Taken from Vision Show – Stuttgart, 2016, public information [66].

2.3 Silicon pixels development

In this section, the three fields that must be studied in order to design, from scratch, a new silicon pixel tracker are listed: silicon design, analogue architecture design, and digital design. Usually, in many scientific collaborations developing a silicon pixel tracker, there is a team of several researchers responsible for each of these design tasks.

Sensor Design

The term sensor design, in this work, refers to the operation of defining the physical structures implementing the actual sensing process in the device; i.e., the creation of the silicon

diode that implements the depleted region. When designing a silicon tracker, there are few parameters that must be taken into account. The main ones are the following:

- Thickness: The thicker the device, the more charge carriers a charged particle creates during its passage. A thicker sensor translates into better efficiency but worse material budget;
- Depletion or diffusion: the free charges can reach the collection electrode either by diffusion or can be forced to do so using a guiding electric field. In the latter case, the time resolution greatly improves, but it is harder to obtain larger sensitive regions;
- Doping: by changing the doping levels of the various layers, as well as their size and shape, it is possible to mould the electric field and engineer the detector performance parameters (efficiency, rate capability, timing and spatial resolution, radiation resistance, etc...);
- Amplification: some sensors provide in-pixel charge amplification to improve the efficiency and the timing performances.

There are a great number of researchers and collaborations developing new ideas and alternatives to well-known paradigms in order to optimise all these parameters and improve the overall performance of silicon pixel sensors. Sensor designers usually make use of complex **T**ecnhological **C**omputer-**A**ided **D**esign (TCAD) software and shape the electric field to maximise the charge collection while minimising interference with the downstream circuitry. In Figure 2.9 a simulation of the cross section of an imaging sensor, in this case ALPIDE one. The structure of the field lines is optimised to grant fast charge collection.

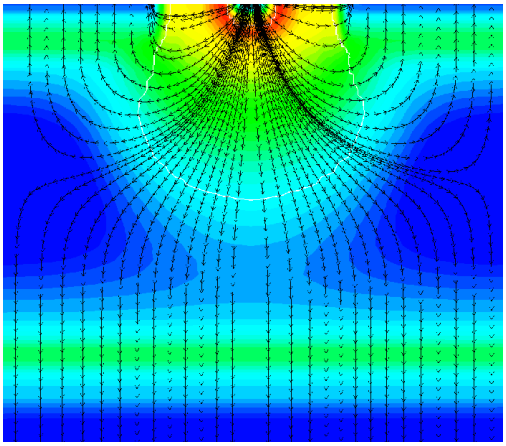


Figure 2.9: A TCAD simulation of ALPIDE pixel. Taken from [67].

Analogue Design

The main goal of the analogue sensor design is to efficiently move charges toward a collection electrode and to make this signal as linear as possible with the energy collected by the silicon diode, implementing what is called the front-end. The analogue front-end stage transforms the charge signal into a voltage signal, amplifies it correctly, shapes it as desired, and in case of a digital output it compares it to some thresholds, thereby creating a digital signal.

Some important parameters for the front end are the shaping time (i.e., the time necessary to produce a valid output signal), the dead time (i.e., the time after the processing of a signal that the front-end is not sensible to further input signals), the gain, and many others. Front-end can be developed either on the same dice where the sensor is or on a different ASIC, but in both cases it is usually implemented in CMOS technology. The transistors connections and sizing determine the performance of such a front-end.

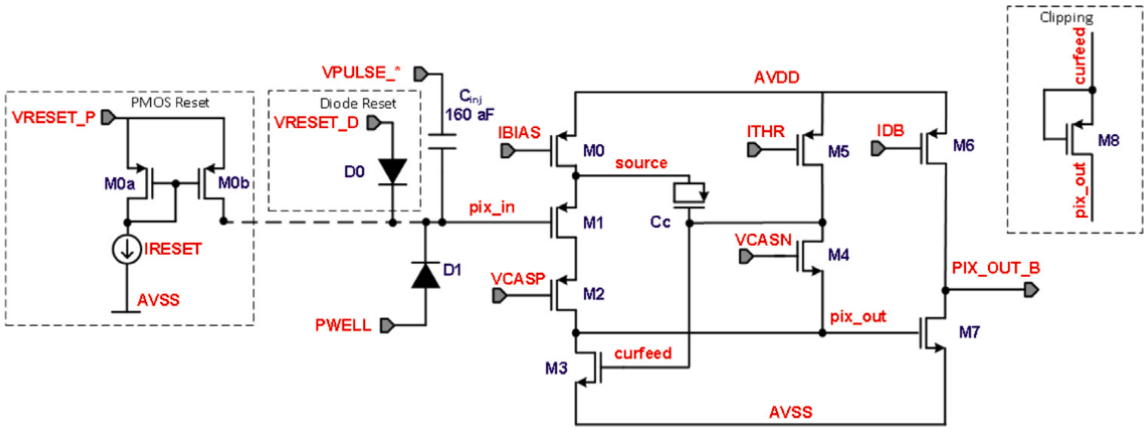


Figure 2.10: The analog circuit used in a MAPS, ALPIDE. Here, reset, amplification, and discrimination are shown. Taken from [10].

In Figure 2.10 part of the analogue design developed for the ALPIDE detector (for ALICE) is shown. In this case, the analogue design is implemented using transistors placed on the same die where detection occurs (ALPIDE is a MAPS).

Digital Design

The digitisation of the signal is necessary to make the data actually readable by a computer-based data acquisition system. This task is performed by an **A**nalog to **D**igital **C**onverter (ADC), that produces a binary output given an analogue signal. ADCs can be implemented in several ways, trading the conversion time with the power consumption and the size. The four main driving principles of ADCs are reported in Figure 2.11. They are the following:

- in **parallel search** (also called flash ADCs) the input signals is compared at the same time with a set of thresholds. The conversion is performed in one clock cycle only. These ADCs are very fast but space- and power-hungry, and are often used in applications requiring very high speed and modest accuracy;
- in **sequential search** a new set of reference level is selected at every new clock strobe. This approach combines high accuracy and rather high speed and is used for many industrial and communication applications;
- in **linear search** the conversion levels are in increaseing or decreasing order; hence the output is monotonically increasing or decreasing. This approach uses the minimum amount of hardware but performs a slow conversion. It is widely used for sensor interfaces;

- in **oversampled converters** the output quickly switches between two states, and the average is an approximation of the actual input. These circuits use only a few reference levels. They are accurate over a large number of samples; they are used in situations where the bandwith is limited and the sample rate can be high.

There also exist hybrid converters using sub–blocks borrowed from different ADC approaches. The technique to pursue, as well as the proper set of thresholds and their spacing, is application-dependent. In a pixel tracker, it is not unlikely for the output of the tracker to be simply binary: a single threshold is set, and a hit is recorded whenever the voltage produced by free charges is higher than this given threshold. This is sufficient to reconstruct the particle trajectory.

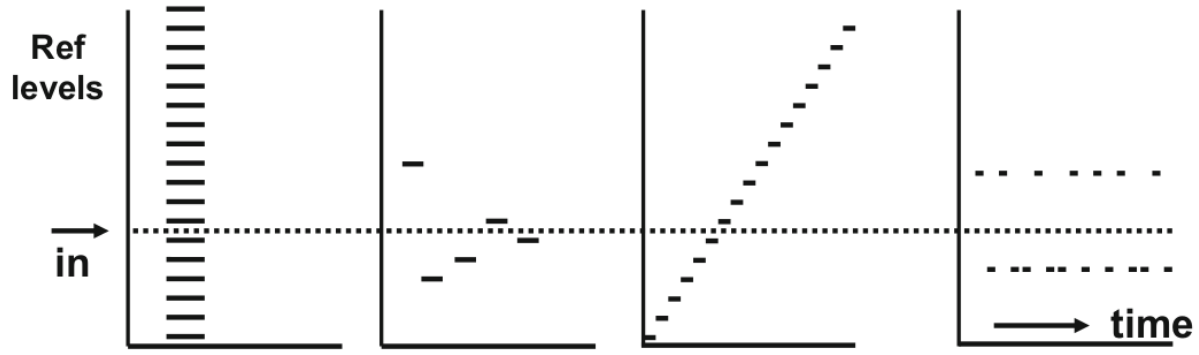


Figure 2.11: The classification of ADCs conversion principles. Left: *parallel search*, middle-left *sequential search*, middle-right *linear search*, right *oversampled correction*. Taken from [68].

During the digitisation process, it is also possible to optimise the signal readout, e.g., implement some zero-suppression mechanisms able to enhance the rate capability by reading only the pixels that are actually containing data. The most recent sensors can also implement directly some data analysis on the sensor itself; one of the main advantages of the microelectronic process used to implement this architecture is its speed, far better than the offline software analysis usually performed on a multi-purpose processor.

Digitisation may happen at different stages of the detector chain: in some devices, it occurs in-pixel, and then the signal is transported as bits along the device itself toward the operator computer. In other devices, the analogue signal is transported, and digitisation is performed right before the communication with a computer. Nevertheless, a digitisation scheme must be defined whenever designing a new silicon tracker; i.e., construct an architecture made up of logic gates capable of performing the correct operations on the signal. In Figure 2.12 an example of a readout scheme for a tracker (LHCb inner tracker) is shown.

This thesis focuses mainly on the development and testing of digital architectures for silicon pixel trackers; an example is the hash architecture thoroughly discussed in Chapter 4 and Chapter 5.

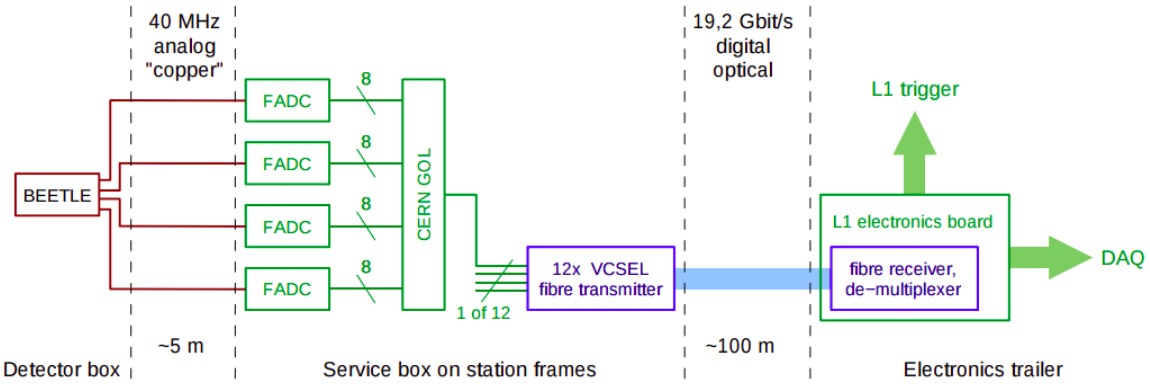


Figure 2.12: The readout scheme for LHCb inner tracker. Here the fast ADCs and the encoding mechanism read out multiple sensors simultaneously. Here flash ADCs (implementing parallel search) are used. Taken from [69].

2.3.1 Techonological node

One of the most crucial decisions, which significantly impacts both the development cost and the overall sensor performance, pertains to the choice of the technological node. When comparing commercial imagers to scientific ones, the nodes used are rarely alike, given that these two domains, which share common technological principles (the transistor working principles), possess distinct requirements and capabilities.

The most evident difference lies in the channel minimal length: whereas the latest digital circuits used in consumer applications go down to 3 nm [70], in the tracking environment the smallest technology node used at the time of writing is 65 nm by an ASIC used for a hybrid device [71] and 110 nm by a MAPS [72].

The main differences between commercial and scientific CMOS sensors in terms of goals, on the other hand, can be summarised as follows:

- Commercial devices usually take frame; i.e., all data in the array are read, and produced data are images or *photos*. Trackers used in HEP take advantage of data sparsification, and only pixels actually containing information are readout, and produced data are usually *positions*;
- Commercial devices are mostly used in the either the visibile spectrum or close regions (infra-red or ultraviolet), where a photon creates a free charge pair in the sensor. Tracker used in HEP focus mostly on charged particles tracking, releasing at least 80 $e^-/\mu m$ [73], therefore usually hundreds (or at least tens, depending on the specific application) of electrons are collected per particle passage;
- Commercial devices employ few relatively small sensors (tens of mm^2) with fixed number of pixels (tens of million). Trackers used in HEP need to use several thousands of sensors covering very large areas (m^2) and maintain a pixel pitch as small as possible.
- The development of commercial devices is driven by a market valued more that USD

20 billion in 2022 [74] and each sensor is used for several million devices. The development of a tracker for scientific purposes is usually tailored over one (or few) experiments and is financed at most for few million USD.

Focusing on the choices driven by the financial constraints, usually when developing a MAPS it is necessary to fine-tune the size of the sensor (having direct impact on the production yield), whether to use a modular approach (repeating the same small sensor) or a stitching one (creating a single big sensor built by physically connecting separate modules) [75], and the technological node that force the price of both the wafers themselves and of the development research necessary to develop a working device. An example of such calculations can be seen in Figure 2.13, where the production yield and the cost per wafer variables are combined to try and keep the cost threshold below 100,000 USD per m^2, a price that can be accepted by scientific research.

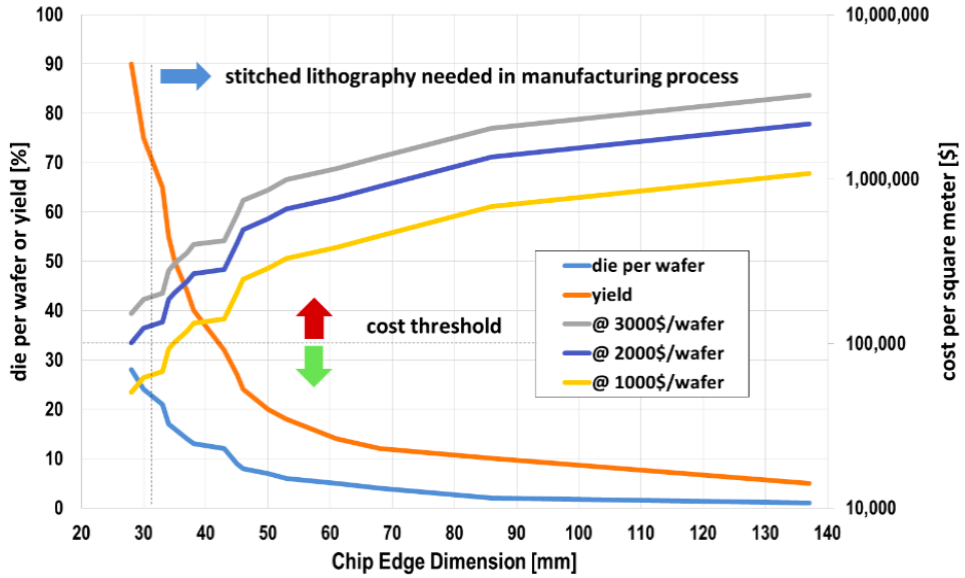


Figure 2.13: Some speculations on the price per square meter of tracker taking into account the price per wafer and the production yield. Taken from [76].

The optimisation of these parameters has paramount effect in the sensor performance: as an example, if the extremely small and power-saving technologies currently used for phone cameras were employed, maybe the need for innovative architectures such as the one discussed here could become optional. However, the current status in MAPS development for scientific applications forces the need for these new ideas to improve the tracking performances. In this work, the technological node (LFoundry 110 nm) is considered fixed and the goal is the optimisation of the overall architecture itself.

2.4 Tracking applications

Tracking is an important part of many scientific experiments. In this section, four of them are briefly summarised to compare the tracking needs and the tracker performances. The four examples are **ALICE** collaboration (an example of HEP), **AMS-02** collaboration (an example of space physics), **PRAvDA** collaboration (for applied medical physics using strips), and **iMPACT** collaboration (for applied medical physics using pixels). iMPACT is discussed in relatively more detail as it was mostly developed in Padova.

2.4.1 ALICE and ALPIDE

ALICE (**A** **L**arge **I**on **C**ollider **E**xperiment) is one of the four main experiments at LHC at CERN [77].

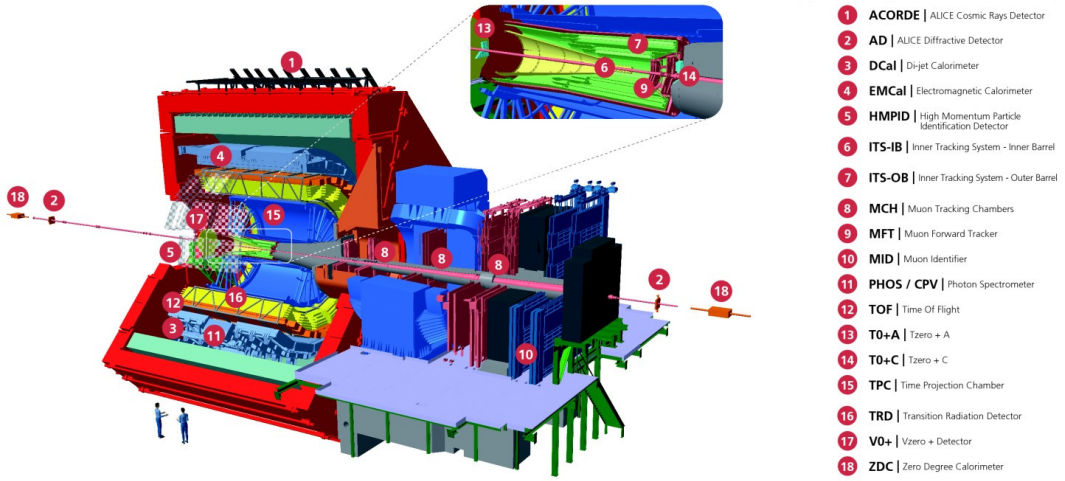


Figure 2.14: A schematic of ALICE detector after the upgrade for Run 3. Taken from [78].

ALICE is a general purpose detector capable of studying both proton-proton and Pb-Pb interactions; it is devoted to the study of the highly interacting quark-gluon plasma created when two very energetic hadrons collide. The structure of ALICE detector resembles the other detectors at LHC with cylindrical symmetry (ATLAS and CMS): it is a barrel where each layer performs a specific task in the detecting chain. A schematic of the detector is shown in Figure 2.14. The innermost layer of the ALICE detector is the ITS (**I**nner **T**racking **S**ystem); the actual particle tracker uses a silicon sensor called ALPIDE [10].

ALPIDE is a (1.5×3) cm^2 large MAPS with 512×1024 pixels that are read out in a binary hit/no-hit fashion (single threshold value). The characteristics of the device are reported in Table 2.1. An image of the sensor can be seen in Figure 2.15.

In ALICE, as in other HEP experiments, some of these quantities are particularly important. Sensor thickness is a trade-off between material budget, detection efficiency, and mechanical stability. Spatial resolution is driven by the technology in which the device is

Parameter	IB	OB
Sensor thickness (µm)	50	50
Spatial resolution (µm)	5	10
Dimensions (mm^2)	15×30	15×30
Power density ($mW\,cm^{-2}$)	300	100
Time resolution (µs)	30	30
Detection Efficiency (%)	99	99
Fake hit rate	10^{-5}	10^{-5}
TID radiation hardness (krad)	2700	100
NIEL radiation hardness ($1\,MeV\ n_{eq}/cm^2$)	1.7×10^{13}	10^{12}

Table 2.1: Characteristic of the ALICE tracker: Inner Barrel (IB) and Outer Barrel (OB). Taken from [10].

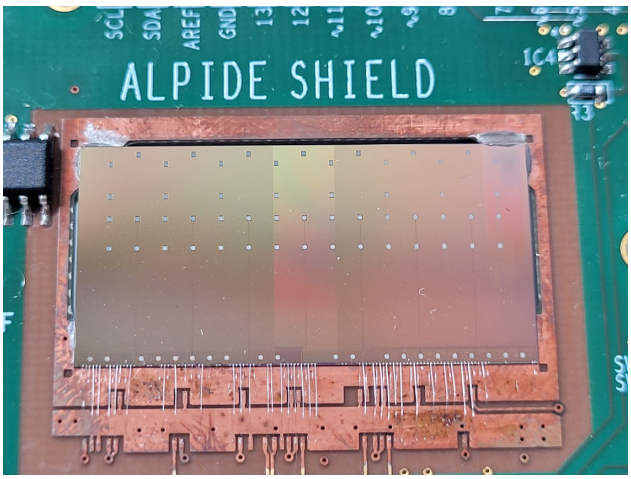


Figure 2.15: A photo of ALPIDE chip mounted on a debug PCB. Taken in Department of Physics and Astronomy of the University of Padova, GRIT laboratory.

developed, the power density budget, and the expected events rate. A bigger pixel translates on one hand into worse spatial resolution, on the other hand on more space to host transistors, hence improving the circuital performances of both the analogue and the digital architectures. Technology, as well as the sensor design itself, also drives the radiation hardness for **T**otal **I**onizing **D**ose (TID) and **N**on-**I**onizing **E**nergy **L**oss (NIEL) effects. If ALPIDE were more power-thirsty, the actual material budget would be higher: the need for more power translates into a more complicated and thicker cooling system. If the radiation hardness were worse, the tracker lifetime would not suffice for a run of LHC accelerator, and the acquired data would not be reliable through the entire acquisition period.

The tracker of the ALICE experiment is very important in the event reconstruction: after the collision, especially in the Pb-Pb case, a plethora of particles, mostly short-lived, start travelling outward toward the ALICE detector. The role of the ITS is to precisely reconstruct the position and the direction of the particles in order to locate the position

of the vertex, the point from which the particle originated, in particular distinguishing the primary vertex from the secondary ones. The ITS is made up of two barrels of silicon tracker, the Inner Barrel (composed of three different layers) and the Outer Barrel (made up of four layers) for a total of about $10\,\mathrm{m}^2$ of silicon [10].

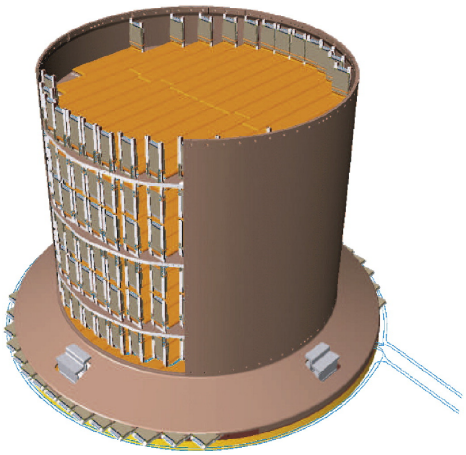
ALICE ITS has been upgraded several times over the years, and the ALPIDE sensor has been used in many scientific experiments beyond HEP because of its great performance. Some more details of the ALPIDE sensor readout architecture, in comparison with other MAPS, is given in Chapter 4; throughout this work it is taken as a reference for such sensors. Currently, the successor to the ALPIDE chip is being developed by the ALICE collaboration; its installation is planned during LHC Long Shutdown 3 (2026 - 2028) [79].

2.4.2 AMS-02

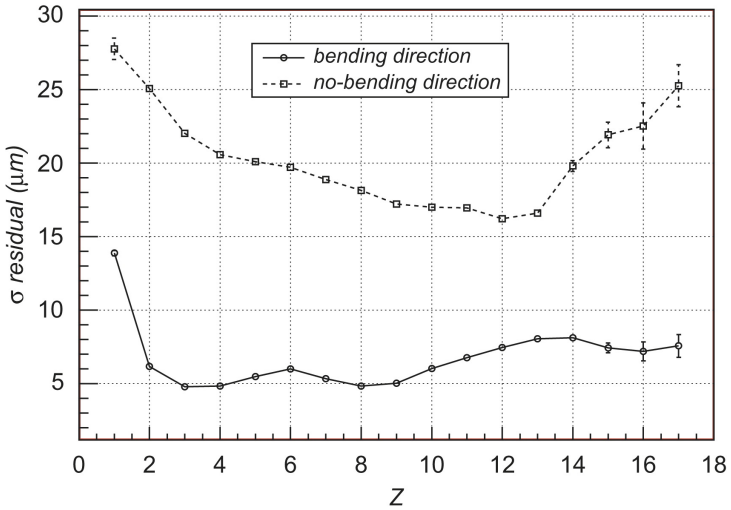
AMS-02 (**A**lpha **M**agnetic **S**pectrometer 2) is a cosmic ray detector designed to operate on the ISS (**I**nternational **S**pace **S**tation). Its scientific goal is to perform precise measurements on the energy spectra of the cosmic rays, looking for dark matter signs. The tracker is made up of eight layers of silicon microstrips operating in a strong magnetic field. The tracker is composed of 2264 microstrips, each with dimensions of:

$$(41.360\ \times 72.045 \times 72.045 \times 0.300)\,\mathrm{mm}^3$$

placed in five support planes, for a total of $6.75\,\mathrm{m}^2$ of sensitive silicon.



(a) A representation of AMS-02 tracker, without the top plane.



(b) Spatial resolution for AMS-02 tracker along the two directions depending on the particle charge.

Figure 2.16: Representation of the work performed for AMS-02 detector. Both figures taken from [80].

The role of the tracker is to reconstruct the radius of curvature R of the trajectory of the particle, thus measuring its magnetic rigidity $BR{=}p/Z$. This information, combined

with the velocity and energy measurements provided by adjacent detectors, can determine the value of the particle charge Z, and hence perform particle identification. An image of the tracker can be seen in Figure 2.16a. This experiment presents a plain example of the importance of spatial resolution: the better it is, the better the particle identification performance. Moreover, as it is placed in outer space, the power constraint is very strict. The entire silicon tracker weighs about 186 kg and consumes only 734 W.

Before the placement of AMS-02 in orbit, it was calibrated using proton, positron, electron, and pion beams at different energies, in order to determine the absolute rigidity scale and the rigidity resolution of the tracker. In Figure 2.16b an image showing the spatial resolution of this tracker can be seen along the two main axes (here named the bending direction and the non-bending direction).

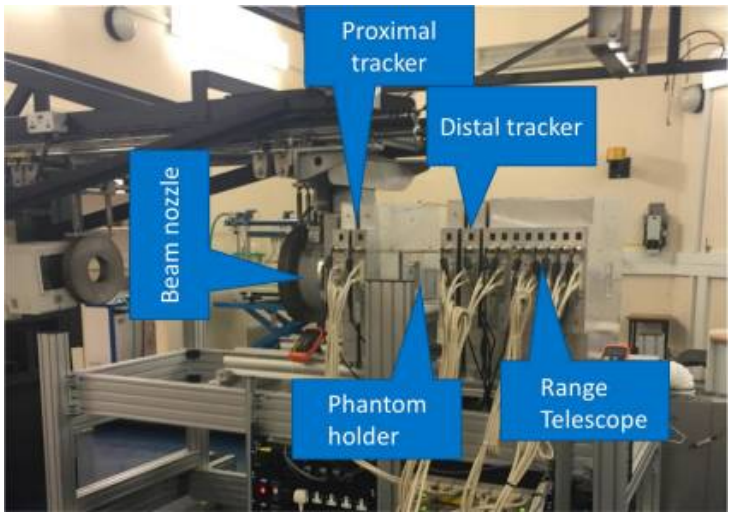
AMS-02 has been flying in the ISS for more than 10 years and has collected, at the time of writing, more than 200 billion cosmic ray events [81] that allowed the collaboration to generate several scientific discoveries and papers.

2.4.3 PRAvDA

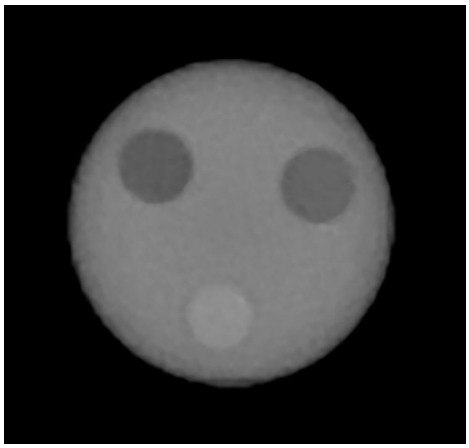
PRAvDA is a consortium working on a solid-state system for **p**roton **C**omputed **T**omography (pCT) [82]. The goal of this experiment is to create a device capable of reconstructing the cross section of biological material (i.e. a patient) to improve hadron therapy, an alternative to conventional radiotherapy, to optimise tumour treatment in fragile locations such as the head or neck. The idea is to use protons as imaging probes: by using a relatively high-energy proton beam (hundreds of MeV) it is possible, combining the information of a tracker and a calorimeter, to measure the particle deviation in a human body, ultimately reconstructing a map of the proton cross section of the patient. In Figure 2.17b a slice of a spherical phantom imaged using the PRAvDA system, shown in Figure 2.17a.

In PRAvDA, individual proton trajectories are tracked using silicon microstrip detectors, 150 µm thick n-in-p silicon with an active area of $93 \times 96\,\mathrm{mm}^2$ and a strip pitch of 90.8 µm. The tracker is read out via custom ASICs developed for this purpose. The sensor used is the same as that developed for the ATLAS experiment at LHC. The calorimeter, downstream, is a range telescope composed of alternate silicon strip detectors and PMMA absorbers. For this application, the tracker must have high detection efficiency, high spatial resolution, low material budget, radiation tolerance, and imaging area. A crucial requirement for clinical application for living patients is a reasonably short duration of the imaging process (maximum tens of s); i.e., the tracker must be able to handle high detection rates and good timing resolution is mandatory in order to follow the particles through the detecting system. In its most recent work, the PRAvDA collaboration claims to be able to perform a scan using 2×10^8 protons/s in their full imaging area (19.2×19.2 mm^2).

Concerning radiation resistance, the RD50 collaboration, on the same detector, measured a charge collection efficiency of $\approx 70\%$ even after a hard irradiation of 1.1×10^{15} high energy protons per square centimetre [83], thus demonstrating the reliability of this sensor.



(a) A photo of the PRAvDA proton tomography scanner prototype.



(b) A pCT slice for a spherical phantom containing 3 substitute inserts (water equivalent, adipose equivalent, average bone equivalent).

Figure 2.17: Results obtained by PRAvDA collaboration. Both figures taken from [82].

2.4.4 iMPACT and ALPIDE

The University of Padova and the INFN of Padova host a project to develop a proton tomography scanner, named iMPACT (**i**nnovative **M**edical **P**roton **A**chromatic **C**alorimeter and **T**racker). This project was funded by the **E**uropean **R**esearch **C**ouncil (ERC) through a Consolidator Grant [85]. The main goal of this project is to address the fundamental limitation that still prevents the pCT technique from being applicable in a clinical scenario: particle rate. The iMPACT scanner is designed to reach at least a few tens of MHz, probably not enough for clinical use, but representing good performance compared to the current state-of-the-art and a concrete step toward the feasibility of the technique[4]. In this project, I contributed in particular to the definition and development of the digital readout architecture, to the study of track reconstructions, and to the physical assembly of the prototype, hence it is described with few more details in comparison to the previous ones. A detailed description of this project can be found in other works [23], [84], [86]–[88].

The iMPACT scanner combines two main sections, each with separate functions, similar to the previously mentioned PRAvDA project but using different technical solutions. iMPACT uses a silicon pixel tracking system to measure the deviation of each proton that passes through the patient and a scintillator-based range calorimeter to estimate the residual energy of the particles. The design choice for iMPACT to use a pixel tracker in place of microstrips reduces the number of silicon layers (2 subsequent planes of pixels are enough to measure both track position and angle, one more layer is necessary if the sensors are based on microstrips) and increases the rate capability. A schematic representation of the layout of the iMPACT project can be seen in Figure 2.18.

[4] A value close to the one published by PRAvDA collaboration.

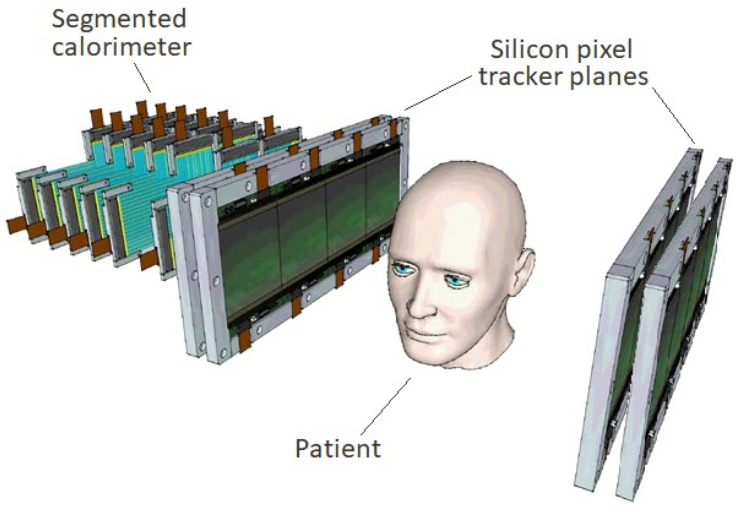
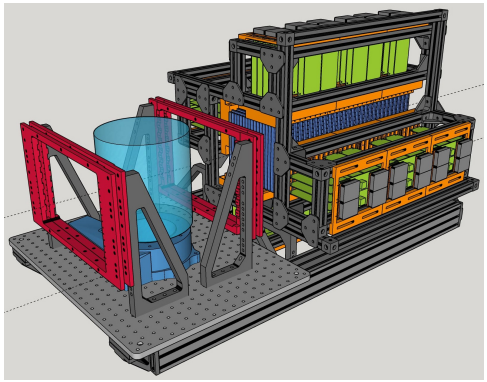


Figure 2.18: Rendering of the iMPACT proton tomography scanner layout. The calorimeter is scintillator-based and a total of 4 planes of silicon pixel sensors are used, two before and two after the patient. Taken from [84].

iMPACT project came to a conclusion in 2021 and produced a working prototype. In Figure 2.19, on the left is a 3D rendering of the scanner, while on the right is the assembled prototype (before the installation of the silicon tracker).



(a) 3D rendering of the mechanics of iMPACT. Here, a commercial phantom, in place of the patient, is used to estimate the scanner performance. Taken from [84].

(b) A photo of iMPACT during the final assembly of the prototype. The silicon trackers were not installed yet. The photo was taken in Department of Physics and Astronomy of the University of Padova, GRIT laboratory.

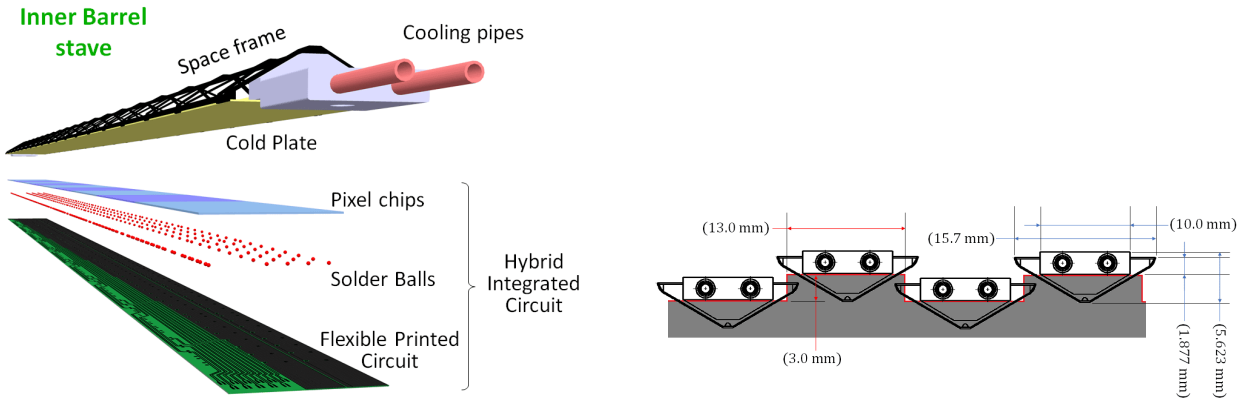
Figure 2.19: iMPACT mechanics.

iMPACT Tracker

In the iMPACT scanner, the tracker is used to measure both the entrance and exit positions and the angle of each particle traversing the scanned object. This information is used to calculate the most probable trajectory of the particles inside the patient, vital for the image

reconstruction. Two tracking planes are placed before the object and two after. Requests on the tracker are:

- The material budget must be relatively low: the more protons are deflected by the silicon planes, the lower the sensitivity on the deviations inside the patient;
- The time resolution must be as high as possible: if the trajectory of particles traversing the patient must be reconstructed, each particle must be tagged and followed through the two trackers and the calorimeter. The higher the number of hits found in the same time frame of the pixel sensor, the harder it will be to correlate the information correctly;
- The space resolution does not need to be too precise: something of the order of 100 µm is more than enough for biological imaging, as the biggest source of uncertainty is given by multiple scattering of the protons inside the body. Nevertheless, the better the precision, the better the resolution;
- The resistance to TID (see Appendix A) due to protons, must be relatively good to allow for continuous operations of the scanner and limit costly maintenance; the reason is that protons in the $70 \div 250$ MeV energy range are $6 \div 2$ times more ionising than MIPs[5];
- The tracking area must be large enough to cover the field-of-view of a patient.



(a) The schematic layout of an Inner Barrel Stave divided in its components. The flexible printed circuit is used to connect with the pads of the ALPIDE sensors mounted directly on the cold plate. Taken from [89].

(b) A side-view representation of the staves mounted on the custom support. Staves are staggered in height. Taken from [84].

Figure 2.20: Some representations on the mechanics of ALICE staves used to create iMPACT tracker.

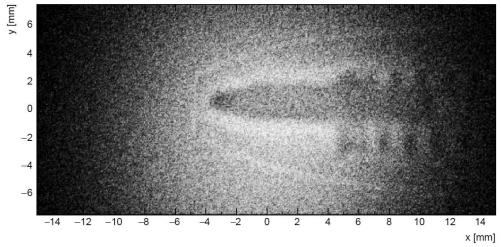
For the iMPACT tracker, the first prototype is implemented using ALPIDE sensors (already introduced in Section 2.4.1). The staves that were developed for the inner barrel of the ALICE detector are used, together with a custom mechanical support to keep them in place allowing for an overlap on the adjacent sensible surface (see Figure 2.20) that, although being small (it is wide about 1 mm), allows the verification of the correct alignment between

[5] A proton is a MIP at ≈ 2.5 GeV [73].

the iMPACT scanner and the proton beam.

The ALPIDE sensor, which was never used before for such a purpose, was validated for proton tomography in an experimental campaign at the TIFPA beamline [90] (Trento - Italy) in 2018 [23]. As a proof-of-concept test, the iMPACT group performed a radiography of a ball point pen shown in Figure 2.21, obtained with a 70 MeV proton beam. Recently, another group, based in Bergen (Norway), used the ALPIDE sensor for pCT [91].

(a) A photo of the ball-tip pen used to try the capability of proton imaging.



(b) The reconstructed image, using 70 MeV protons. Grey scale is not linear for enhanced clarity of the picture.

Figure 2.21: The proton radiography of a pen performed with ALPIDE sensor. Taken from [84].

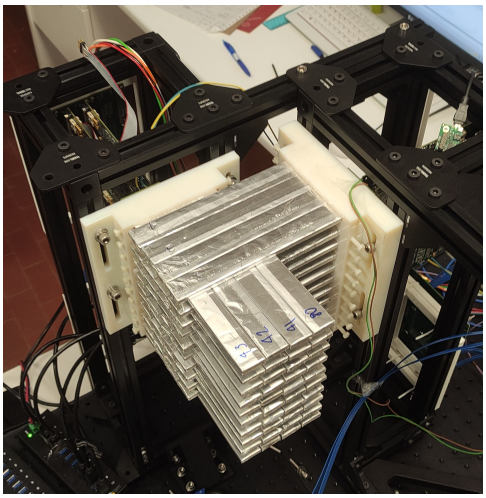
Although ALPIDE can work for pCT, the iMPACT group started considering a different MAPS that could better satisfy the needs of the proton tomography scanner. The sensor developed in this framework is the ARCADIA sensor, capable of handling high rates (up to $100\,\mathrm{MHz\,cm^{-2}}$), which is described in Chapter 3. In fact, the bottleneck limiting the tomography total time is the timing resolution: if there are too many hits with the same timestamp frame, it is hard to correlate the hits in the four tracking planes correctly.

iMPACT Calorimeter

The calorimeter, developed in house, is a range calorimeter: it is used to determine the energy of a proton by measuring its range in a known material (plastic scintillator). It is made up of 240 scintillating fingers 0.5 cm thick, 1 cm wide and 20 cm long, arranged in 60 scintillating planes made of 8 fingers each to create a compact sensitive volume. In this first prototype, the sensitive area is limited to 8×8 cm^2; nevertheless, the same mechanical structure can host a larger version with a sensitive surface of 16×16 cm^2. This calorimeter can measure proton energies up to 230 MeV, enough to perform proton tomography of a human patient ($\approx$ 30 cm water). Along the beam direction, the planes are alternatively rotated by 90 degrees: even-indexed planes are along the x axis and odd-indexed one along the y axis (see Figure 2.22b).

Each scintillator is wrapped in a thin, highly reflective teflon layer to increase the light collection efficiency and surrounded by a single aluminium foil, to block any external light source and prevent optical crosstalk between adjacent scintillators. Each finger is singularly readout by a **Si**licon **P**hoto **M**ultiplier (SiPM) placed on one of the end surfaces. After proper amplification and shaping, the analogue signal is digitised via comparison with

(a) A photo showing a complete module for the x direction. Each SiPM reads a single scintillating finger. A module reads 40 scintillators, all along the same axis, organised as a 4×10 array.



(b) The same setup after both x and y modules are completely installed forming a total of 20 planes: 10 along the x axis and 10 along the y axis. Aluminum wraps enclose two different scintillators to limit the material budget.

Figure 2.22: Photos taken while mounting iMPACT calorimeter. Taken from [84].

three programmable voltage thresholds for each channel, providing a coarse estimation of the energy deposition in that particular finger. The digital signals from all the scintillators that form the calorimeter are digested by a DAQ system developed using Xilinx **F**ield **P**rogrammable **G**ate **A**rrays (FPGAs) [92]; the 8×8 cm^2 calorimeter prototype includes a total of 12 FPGAs. For each of these signals, a timestamp is associated: this is essential in order to correlate hits in multiple fingers into individual particle tracks during the analysis and image reconstruction phase. The timestamp is based on a 100 MHz clock distributed to all FPGAs, ensuring a timing resolution comparable to the light collection time inside the scintillator $O(10\,\text{ns})$. Some photos taken during calorimeter construction can be seen in Figure 2.22.

Calorimeter segmentation has been chosen to optimise scanner performance: the pitch along the depth is chosen to obtain a range measurement precision (estimated spatial resolution $\sigma_z = \Delta z/\sqrt{12} \approx 1.4\,\text{mm}$) comparable with the intrinsic statistical fluctuation given by range straggling (around 1.1% at $\approx 200\,\text{MeV}$, that is, $\approx 3\,\text{mm}$), while the segmentation in the transverse plane can sustain higher particle rates and provide coarse information on the particle trajectories.

Figure 2.23 shows one of the preliminary cosmic ray tests (muons) performed to check the correct functioning of the readout logic of the calorimeter: here, three instances of muons passing through the scintillating fingers are reported as seen by two different FPGAs that also register the energy deposit (represented by the different colour). To obtain these images, the two FPGAs must be properly synchronised.

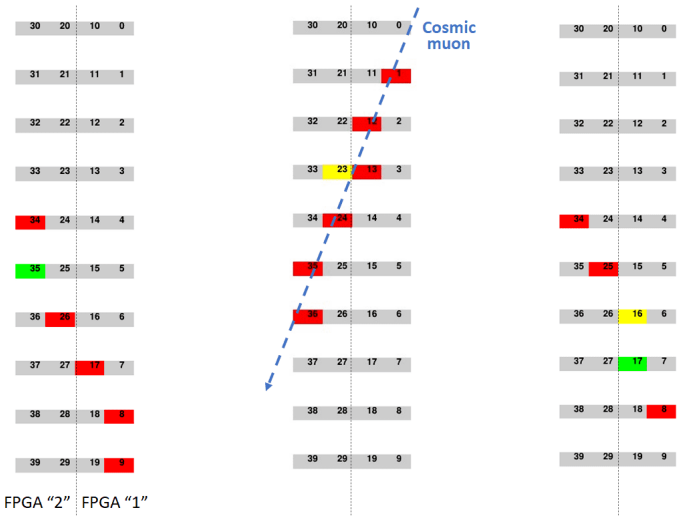


Figure 2.23: Three different muons recorded with the calorimeter with a total of 40 scintillators, read out by two different FPGAs. A possible muon track is superimposed in the middle one. Taken from [84].

iMPACT results and future perspectives

The feasibility of the iMPACT project was demonstrated and a working prototype was created. The tracking sensor was validated with a published article [23], while the calorimeter (mainly due to Covid-19 restrictions during 2020 and 2021) was validated using cosmic rays (see Figure 2.23). At the time of writing, a paper, soon to be submitted for publication, is being worked on. Preliminary results can be found in dr. Baruffaldi's Ph.D. thesis [84].

According to the Monte Carlo simulations [84], the iMPACT scanner should be able to perform the pCT of a biological phantom within two minutes. A full test with a medical proton beam will be performed as soon as possible. Then, the emphasis will shift to developing a proper reconstruction algorithm in order to obtain a full tomographic 3D image.

Chapter 3

ARCADIA

ARCADIA (**A**dvanced **R**eadout **C**MOS **A**rchitectures with **D**epleted **I**ntegrated sensor **A**rrays) is a project funded by INFN's CSN5 (The Technology, interdisciplinary, accelerator development, and electronics commission of the Italian Institute of Nuclear Physics) that aims to develop a multi-purpose FD-MAPS (**F**ully **D**epleted MAPS) [25] using a modified LFoundry 110 nm commercial CMOS process [30]. This project numbers contributors from many sections of INFN (Bologna, Milano, Padova, Pavia, Perugia, Torino, and TIFPA in Trento) [93]. I had the opportunity to actively participate in the collaboration, contributing both in the preliminary validation campaign, in the acquisition system development, in the radiation hardness study, and in the experimental campaign. Within this project framework I also developed the hash readout architecture, which is thoroughly discussed in chapters 4 and 5.

This chapter begins with an overall presentation of the project and of the work carried out by the collaboration (Section 3.1). After this section, the data acquisition system developed for the experimental tests will be presented together with a few plots produced with radioactive sources (Section 3.2). In the last Section 3.3, an experimental study on the sensor tolerance to non-ionizing radiation, performed for the sensor commissioning, is shown.

3.1 Project Overview

Three engineering runs are planned during the ARCADIA project. Each engineering run implemented both a **M**ain **D**emonstrator (MD) and a set of smaller test structures. Personally, I had the opportunity to study the performance and use ARCADIA-MD1, the main demonstrator of the first engineering run; hence, the work presented here refers to this sensor. The collaboration decided not to conduct a thorough experimental campaign with ARCADIA-MD2 due to some issues that made its usage delicate; ARCADIA-MD3 has been produced and is currently under preliminary tests, showing great performance.

It was decided to create a digital device with a single threshold; hence the energy collected

by the diode is not propagated to the user but data consist in the addresses of the hit pixels. This choice was done in order to keep the power consumption as low as possible and allow for small pixels.

3.1.1 Sensor applications

ARCADIA sensor is multipurpose and the different applications impose requirements on the sensor performance. In particular, the main applications envisioned are the following:

- High-energy physics: ARCADIA is one of the candidates for a tracker in the future circular lepton collider. To make this sensor viable, it is particularly important that it maintain low power consumption (below $40\,\mathrm{mW\,cm^{-2}}$), be able to maintain particle rates of tens of $\mathrm{MHz\,cm^{-2}}$, and have at least low to medium radiation tolerance (up to $\approx 10\,\mathrm{kGy}$ TID).
- Medical applications: as discussed in Section 2.4.4 ARCADIA is a candidate for a proton tomography scanner. This request adds, to the already mentioned requirements, an even stricter constraint on the material budget and a very large surface area, ideally around $20\,\mathrm{cm^2}$. A particle rate capability as high as possible would help making pCT feasible by reducing the duration of the patient exam;
- Space applications: in order to use ARCADIA in space the requirement for power consumption is even stricter. The goal set was below $10\,\mathrm{mW\,cm^{-2}}$.

A summary of the ARCADIA sensor parameters chosen as the goals of the project is presented in Table 3.1. These constraints have guided the collaboration to optimise as much as possible the produced device from the sensor design, analogue architecture, and digital architecture points of view.

Parameter	Minimum specification	Maximum specification
Sensor thickness	$50\,\mu\mathrm{m}$	$200\,\mu\mathrm{m}$
Pixel pitch	$25\,\mu\mathrm{m}$	-
Array area	$1.28\,\mathrm{cm} \times 1.28\,\mathrm{cm}$	-
Power consumption	$10\,\mathrm{mW\,cm^{-2}}$	$20\,\mathrm{mW\,cm^{-2}}$
Hit rate	$10\,\mathrm{MHz\,cm^{-2}}$	$100\,\mathrm{MHz\,cm^{-2}}$
Timing resolution	$\mathrm{O}(1\,\mu\mathrm{s})$	$\mathrm{O}(10\,\mu\mathrm{s})$

Table 3.1: Goal requirements for ARCADIA sensor, set by the collaboration before the start of the device development.

3.1.2 ARCADIA development

Sensor design

FD-MAPS, compared to standard MAPS, have improved sensor radiation tolerance and speed due to drift-dominated charge collection. ARCADIA relies on a standard CMOS production process, enhanced by a backside junction implanted through a patented low-temperature processing [25]. The collection electrodes are located on an n-type epitaxial layer grown on top of a high resistivity n-type substrate, and the CMOS circuits are hosted by the deep p-well implanted around the collection electrode (see Figure 3.1). Depletion starts from the backside of the device; thus, signals are hardly visible until the device reaches full depletion.

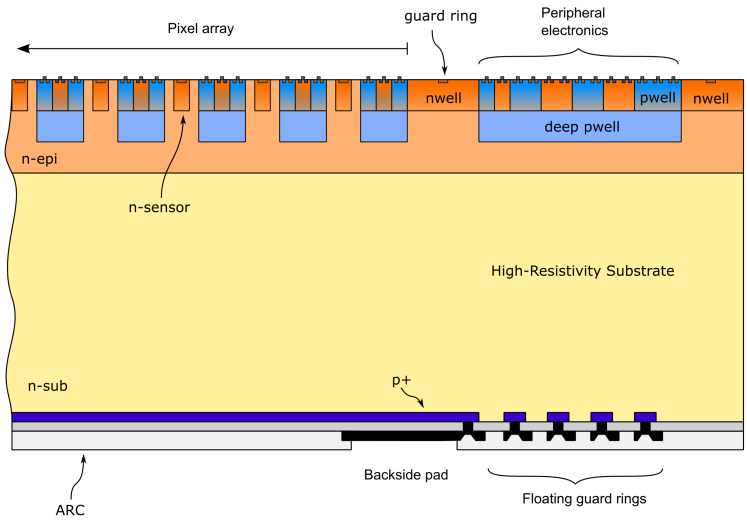


Figure 3.1: A cross-section of the ARCADIA sensor as designed. Taken from [25].

Analogue architecture development

ARCADIA collaborators took advantage of the commercial process to experiment with different flavours for the analogue architecture. They explored various possibilities for the pixel front-end (to amplify the charge and extract linear voltage information) and for the serializers in the periphery, among other things. During the various engineering runs, the collaboration produced several test structures and flavours of the Main Demonstrator to evaluate the performance of the different feasible ideas.

Digital architecture development

A schematic representation of how the ARCADIA-MD1 sensor is divided hierarchically can be seen in Figure 3.2. The device is composed of two major regions: the clocked periphery, where all the modules needed for data retrieval, power distribution, and serialisation are

present; the clockless array, containing all the collection diodes that retrieve the electric signal generated by the ionising radiation. The array is divided in 16 sections, 512-pixels high and 32-pixels wide. Each section is divided into 16 columns, only 2-pixels wide. Each column is then divided into cores, each core implements 4 pixel regions made of 8 pixels each, organised in an array 2×4. A clustering algorithm is implemented: pixel regions are always read as a single entity, and the hitmap of all 8 pixels in the pixel region is communicated to the periphery whenever there is at least one hit. Each section communicates its data via a different serializer; therefore, there are 16 different serializers communicating data at the same time.

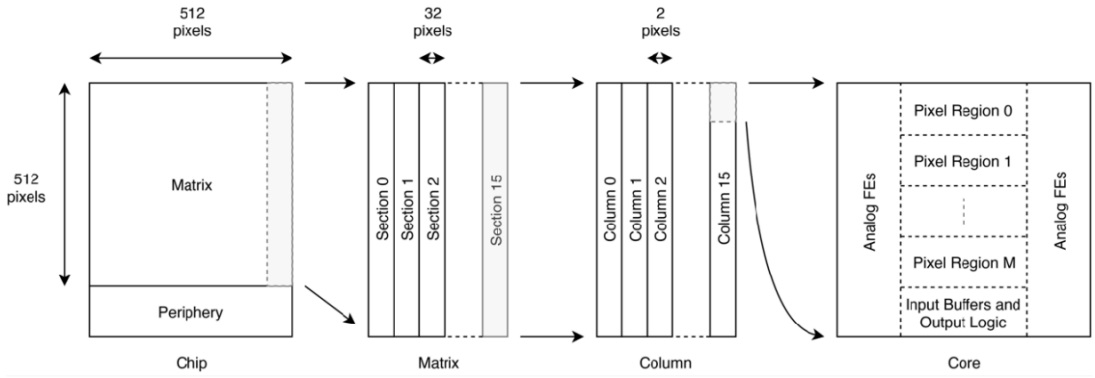


Figure 3.2: A schematic representing ARCADIA-MD1. Taken from [93].

During the various engineering runs, minor changes and fixes were implemented in the architecture to improve the performance of the device and fix bugs spotted during verification and testing, but the structure just described was preserved.

Regarding the transmitted data, the ARCADIA sensor provides both spatial and timing information of the impinging particle. Spatial information is given by the sector address, the column address, the pixel region address, and the hit map of the pixel region. Timing information is retrieved through a timestamp generated by a counter running in the periphery. This value is copied when the hitmap coming from a pixel region in the array reaches the periphery. By default, this counter has 8 bits.

The chip, through the 16 serializers in the periphery, communicates serially 32-bit words, organised as follows (starting from the **L**east **S**ignificant **B**it - LSB):

- 1 dummy bit used for a particular type of readout that will not be discussed in this work;
- 8 bits for the one-hot encoded hitmap of the pixel region;
- 7 bits for the address of the pixel region inside the double column;
- 4 bits for the address of the double column inside the section;
- 4 bits for the section address inside the array;
- 8 bits for the timestamp

A representation of such a word can be seen in Figure 3.3. These words are passed bit by bit through one of the serializers in the periphery in `8b10b` encoding (hence 40 bits are

transmitted per word) [94]. If there are no data to transmit, a 32-bit synchronisation word is written (always `8b10b` encoded) to maintain synchronisation between the words written and their sampling.

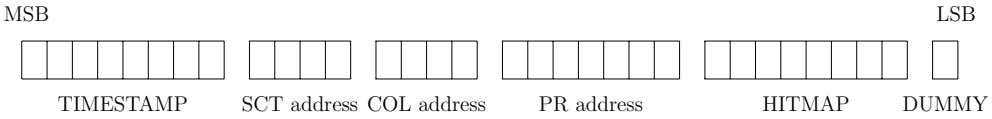


Figure 3.3: A data word (32-bit) coming from an ARCADIA MD1. HITMAP is one-hot encoded, while all other parts of the word have binary representation.

3.1.3 Verification

Both before and after the engineering runs, a lot of effort was put into verifying the chip under development. A significant amount of Monte Carlo simulations were utilised in this process and are summarised here.

Sensor

From a sensor point of view, simulations were performed using the Synopsys Sentaurus TCAD tool [95]. These simulations were used to predict many quantities including (not exhaustive list) the depletion voltage, the onset of the punch-through voltage, the collection speed, and the charge sharing among adjacent pixels. In some cases, other Monte Carlo tools for particle transport in matter (Geant4 [96]) were used to estimate the charge deposit in the device. The simulation domain ranged from a single pixel to bigger arrays made of tens of pixels, and also considered the production process corner cases to verify the design stability. Simulations are still ongoing, but some papers have been published describing the results obtained with these simulations [25], [97], [98].

Analogue architecture

The development and validation of the analogue architecture involved intensive simulations (using SPICE-like software) to evaluate the performance of the architecture by changing the transistors sizing and biasing. These simulations were used to identify the best choices for the architecture and to estimate critical parameters such as the dead time of the front-end after readout and the overall power consumption.

Digital architecture

Digital architecture simulations, just like the analogue ones, were used both to validate the chosen architecture and to anticipate sensor performance in various environments. They were all written in SystemVerilog [99], like some of the simulations discussed in Section 5.3

and require, as input, some parameters that can be extracted from both the sensor simulations and the analogue architecture ones (such as the collection speed and the front-end dead time). The first simulations developed were the functionality simulations, where various phases of the device functioning were recreated to check whether the architecture was stable and behaving as expected (the startup, the configuration routine, the arrival of a hit, data transmission, etc.). In addition, realistic simulations were performed in the target environment of this sensor. In particular:

- uniform distribution of MIPs with different incident angle spectrum;
- uniform distribution of particles in a space-like environment;
- uniform medical-like proton beam;
- bunched beam with different incident angle and bunch timing;
- collimated beams that could be obtained in an experimental room.

To make the simulations run in a reasonable amount of time, many approximations were made. The results are preliminary and will be published only after an experimental campaign, but the sensor demonstrates very high efficiency (more than 99.5%) even for particle fluxes of the order of $100\,\mathrm{MHz\,cm^{-2}}$.

3.1.4 Data Acquisition system

The term **Data Acquisition system (DAQ)** refers to all the accessory development needed to successfully operate and characterise the device experimentally. In the case of ARCADIA, this included the development of the ancillary hardware, the development of the firmware to run on the FPGA, and the software to run on the computer for proper communication and visualisation. The software development, where I was heavily involved, is discussed in Section 3.2. In Figure 3.4, it is possible to see a schematic representation of how the sensor must be connected to the computer for a laboratory test.

Hardware components

From a hardware point of view, three complementary boards (in addition to the sensors) are necessary according to the ARCADIA design: the **F**ront **E**nd **B**oard (FEB), the **B**reakout **B**oard (BB) and a commercial Xilinx evaluation board (KC705) that hosts a Kintex-7 FPGA [92]. Both the FEB and the BB were designed in-house, respectively, by INFN Torino and INFN Bologna. The FEB is a **P**rinted **C**ircuit **B**oard (PCB) where a single ARCADIA sensor is glued and bonded (i.e. a $12.8 \times 12.8\,\mathrm{mm^2}$ device), while the BB is a PCB used as an interface between (up to two) FEBs and a KC705. In the KC705 there is enough processing capacity to control a total of three different sensors; so a full setup (according to current DAQ development) is composed of a single KC705, two BBs, 3 FEBs and 3 ARCADIA-MD1 chips (see Figure 3.4).

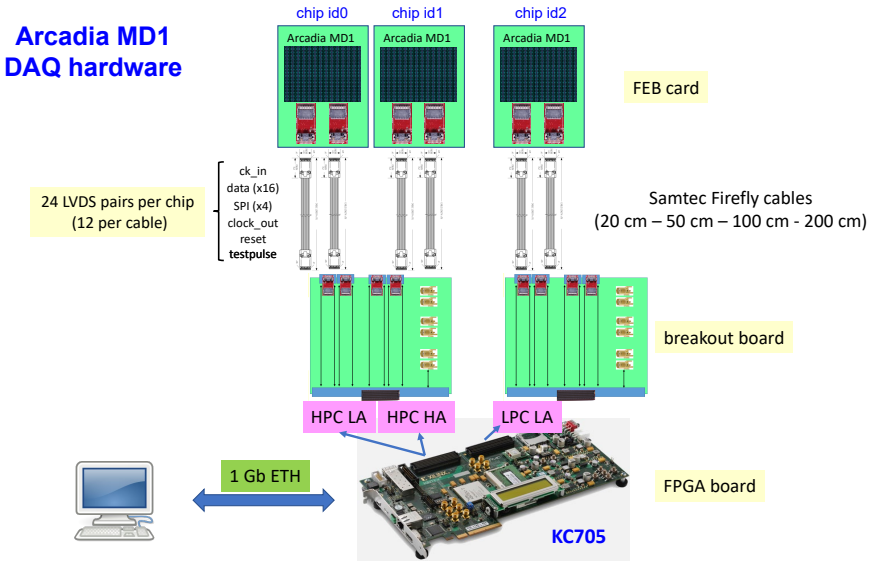


Figure 3.4: A schematic representation of ARCADIA DAQ. Courtesy of dr. D. Falchieri.

Firmware development

Firmware was developed by INFN Bologna. Here, the basic of the protocol used by the readout chain is reported together with the set of messages that can be succesfully read from the experimental setup.

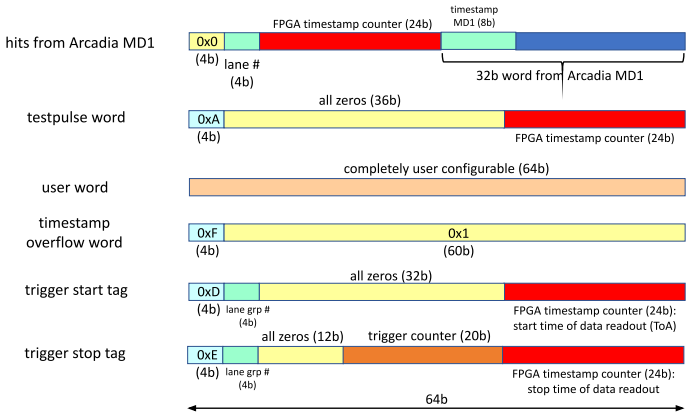


Figure 3.5: The set of words coming from the FPGA. The first 4 bits identify the type of word. Courtesy of dr. D. Falchieri.

The 16 outputs of the MD1 serializers are read at the same time. Whenever a word from the sensor reaches the FPGA, a longer timestamp is immediately associated to the word to keep track of the moment when this word was read. This timestamp, which has the same

reference clock as the one coming from the chip, is longer, as it is made of 24 bits, hence overflows far less often. Then, this FPGA word is queued in a **F**irst-**I**n-**F**irst-**O**ut memory (FIFO) inside the FPGA that is read by the computer through an Ethernet connection. Different types of FPGA words are possible; all of them are 64 bits words and can be recognised by looking at the first 4 bits starting from the **M**ost **S**ignificant **B**it (MSB). All of them are depicted in Figure 3.5. The presence of two separate timestamps (one from the sensor and the other from the FPGA) forces the need for an operation called **timestamp synchronisation**. As this operation must be performed via software, it is discussed in the next section.

Figure 3.6 shows a timeline representing the data from their creation within the array to their transmission to the computer. Only some steps have been listed to provide an impression of the complete readout chain.

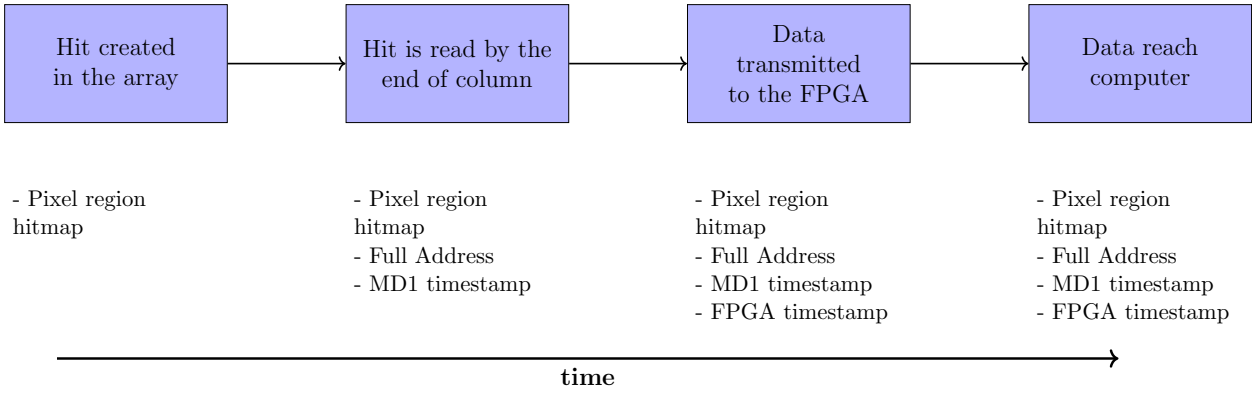


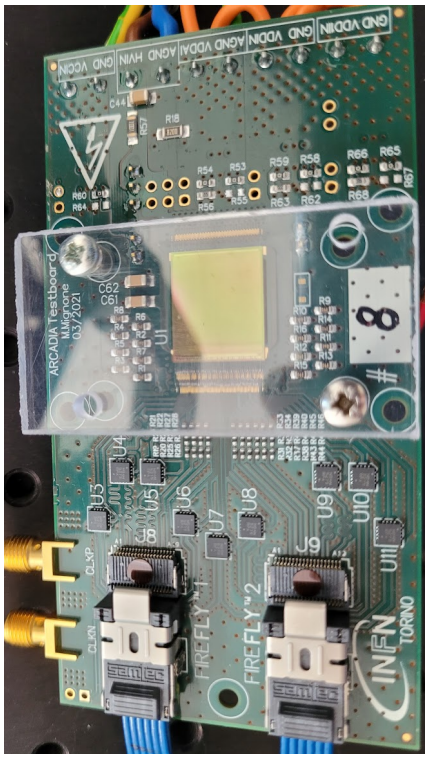
Figure 3.6: Timeline of the data creation process. Below each step, the current information known so far is listed. Every step might be delayed due to high hit multiplicity.

Concerning the driving of multiple ARCADIA chips by the same FPGA, in the current implementation the sensors are almost completely independent: data are saved on separate memories on the FPGA, the timestamp is taken from different counters, and all the firmware parameters are independent for each sensor to ensure maximum operational flexibility. There are only two signals shared among the three readout lines: the testpulse signal (which is propagated at the same time to any connected device) and the timestamp reset signal (so each timestamp counter in the FPGA is reset at the same time).

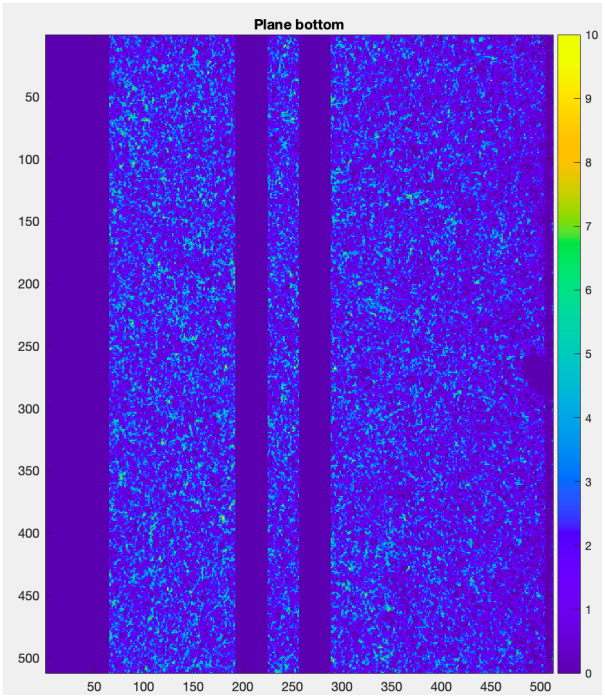
3.1.5 Ongoing work and future perspectives

Although ARCADIA-MD3 is being distributed to the collaboration, some laboratory tests are still being performed on ARCADIA-MD1 and more are envisioned in the near future. The sensor was validated using different particles: cosmic muons, alpha particles from ^{241}Am, electrons from ^{90}Sr, and X-rays from ^{55}Fe. The collaboration is currently working on testing the first hodoscopes of sensors in order to estimate MD1 efficiency and spatial resolution. In Figure 3.7a an MD1 on its front-end board, while in Figure 3.7b a preliminary plot obtained

using ^{90}Sr. The next section will show some further details, as well as other experimental plots.



(a) A front-end board with one of the ARCADIA-MD1 mounted.



(b) A preliminary image taken with ARCADIA-MD1 using with a ^{90}Sr source. Here, the four blank sectors are due to problems during the production of the sensor. Courtesy of prof. R. Santoro.

Figure 3.7: Some images of the ongoing experimental campaign with ARCADIA-MD1.

Among the ARCADIA activities not described yet in this section, there is an alternative digital architecture, called hash architecture, which is the main focus of this thesis and is discussed in Chapter 4 in comparison with other digital architectures. The need for an alternative digital readout architecture is pushed by two limiting factors of the current ARCADIA one: the power consumption and the pixel pitch. Although these values look extremely promising (for example, a power consumption of 10 mW cm^{-2} would be the lowest in a MAPS for charged particles tracking), some experiments lay stricter restrictions (power consumption threshold for space applications is usually set to 5 mW cm^{-2}; concerning the pixel pitch, ALICE3 tracker upgrade [100] asks for pitches around 10 µm). The hash readout architecture, thanks to its simple in-pixel logic, can be used to shrink the pixel pitch in comparison to the current ARCADIA-MD1 pixels; moreover, if the readout routine is pushed toward its limits, the sensor would be less power consuming (as discussed in Section 5.4). Currently, the hash readout architecture has been validated only in simulation.

3.2 ARCADIA Data Acquisition

The purpose of this section is to describe the software readout system developed for the ARCADIA sensor as guided by the physics applications envisioned; it must be thought of as an example of the necessary development effort in order to obtain a sensor that can be used for scientific research. The work presented here refers specifically to the readout of ARCADIA-MD1, nevertheless a very similar readout chain can be used to read a sensor implementing the hash architecture presented in the next chapter; many ideas for the sensor interface were borrowed from this sensor. Personally, I participated in the design, development, and debugging of the ARCADIA readout routine from a software point of view; hence this section is separated and more detailed in comparison with the previous ones. I also participated in the commissioning campaigns (by using both Monte Carlo and experimental methods) for the sensor itself. The backbone of the software described here and the main framework have been written by researchers in INFN Torino and INFN TIFPA (Trento). Here, the functioning of the code is presented together with a few preliminary experimental results.

In the first Section 3.2.1, the driving principles that led to the development of such a software are shown; in Section 3.2.2 the techniques to implement the low-level communication with the hardware are unfolded; in Section 3.2.3 the actions needed to start the device functioning are listed; in Section 3.2.4 a description of how to use the software to take data from a single device is reported; finally in Section 3.2.5 the software capability to acquire data from multiple devices at the same time is presented. This software is actually in continuous development: some of the features discussed at the time of writing might be improved, extended, or even removed in the future versions.

3.2.1 Software principles

An ideal DAQ software for a scientific research device is meant to be used by *users*; i.e., collaborators focusing on the response of the device used and not on the development and implementation of communication paradigms. It should have the following characteristics:

- **Completeness**: the software should allow the user to perform every action that comes to mind to try and stress the device under study;
- **Easiness**: the software should allow the user, a non-developer, to interact with it without leaving his comfort zone;
- **Clarity**: the internal software procedures should be clear to everyone using it, not only to those who developed it;
- **Portability**: the environment that the software relies on should be as general as possible, as different users may want to use a different **O**perative **S**ystem (OS) or rely on different libraries.

There is a plain trade-off between all these characteristics, as the more complete a software becomes, the less easy and clear it becomes. It is exactly for this reason that modern software relies on layered architectures [101] where the most fundamental actions are performed internally and the user has access to simple methods that conceal the actual underlying

complexity.

ARCADIA DAQ software, simply named **arcadia-daq-sw**, has been developed to achieve the following goals necessary to use the experimental setup:

- Open the communication channel between the user's computer and the actual physical device, i.e., the FPGA and the chip itself (through the FPGA);
- Bring both the chip and the FPGA to a default state where the behaviour of the system is known and predictable;
- Test the response of the system to check if everything is working properly;
- Set-up the correct parameters for the system to work as desired;
- Start data acquisition with up to three devices at the same time;
- Show acquired data online to allow fast fixes;
- Save acquired data in a format ready for offline analysis.

Some configurable parameters also allow for a customisation of the experimental campaign. In particular, it is possible to enable:

- Triggered acquisition (a physical trigger signal is needed in the FPGA);
- Real time clustering and cluster analysis;
- Real time erasing of *blob-like* clusters;
- Real time auto-masking of noisy pixels;
- Multi device acquisition with online recognition of synced data.

These routines are described in detail throughout this section.

Arcadia-daq-sw has been written using two different languages: the back-end (communication with the hardware and data readout) is implemented in C++ [102] using CERN IPbus suite [103], while the front-end is developed using Python3 [104] and, in particular, MatPlotLib [105] library for data visualisation, and NumPy [106] for data management.

3.2.2 Low-level communication

CERN IPbus software suite implements a protocol to read and modify registers in an FPGA-based design. This library is used to allow changing the set-up parameters on either the chip or the FPGA, reading them, and asking the chip for data packets.

The scheme in Figure 3.8 shows how communication is implemented using two different interfaces that interact with the DAQ: one for the chip (`Chip IF`), and one for the FPGA (`FPGA IF`). As shown, communication between the user and the hardware takes place only inside `Chip IF`. This is due to the fact, anticipated before, that different portions of the FPGA implement communication with a given ARCADIA sensor: when a parameter must be set on the FPGA, it is necessary to instruct the acquisition system to change it in the region linked to the correct chip.

Whereas data stream and test pulse stream are unidirectional, the setup stream is represented as bidirectional, as it is possible to both read and write in registers inside the FPGA

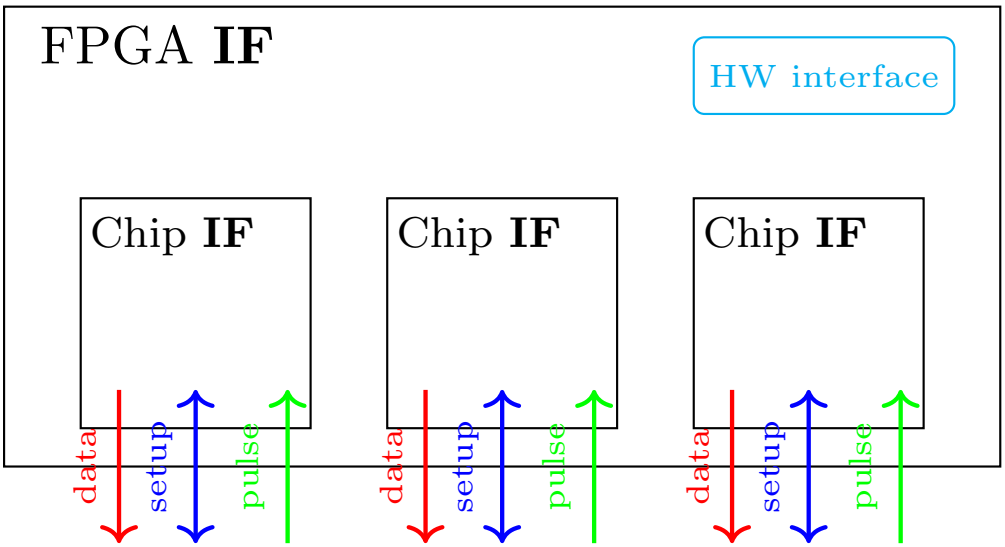


Figure 3.8: Back-end communication of arcadia-daq-sw; **IF** stands for **interface**. Every `FPGA IF` contains three `Chip IF`, as each FPGA can drive up to three chips. The interface with the FPGA contains the IPbus suite interface to communicate with the hardware. Communicating with the chip interface it is possible to read data, read and set acquisition parameters, and send test pulses.

or the chip itself. Here, some of the parameters that can be set on the side of the FPGA are listed together with the issue that pushed the need for such a configuration parameter. Many of these will be used in the next sections describing the routines that the DAQ system needs:

- Electrical signals in the communication between the FPGA and the sensor can be delayed due to instrumental effects, therefore, data packet sampling (through serializers) can fail because of either hold or setup violations (sampling occurs close to a signal transition [107]). To solve this, it is possible to add some delays to the outputs of the serializers, changing the sampling phase (this will be used for the deserializers calibration, explained below);
- The FPGA communication is performed via `8b10b` encoding. Issues in communication can be spotted in the decoding phase, due to the very same instrumental effects mentioned before. Therefore, these errors are counted in the FPGA, and the value can be retrieved and reset;
- The timing precision of the tracking sensor is not known *a priori*, as the overall uncertainty may be different to the simulated one. Considering that ARCADIA-MD1 is a prototype, the possibility to set the timestamp width as a number of reference clock cycles was allowed. This is needed also in timestamp calibration routine, explained below;
- Some serializers do not work properly because of production problems. To avoid being flooded with meaningless data packets, it is possible to disable the readout of some of them;
- ARCADIA-MD1 is a sensor in data push mode (data are always broadcast toward the

FPGA as soon as they are available); the FPGA can work in trigger mode. This mode can be enabled via a proper configuration register, and its parameters can be set (more details on the trigger mode are explained below);
- During the experimental campaign, some recurrent noisy words coming from the sensor were found, probably due to some production problems. To solve this, some configurable masks to filter them were created;
- The three FPGA timestamp counters (one per chip) are used to extend the sensors timestamps, and they are independent of each other. Therefore, it is necessary to reset the FPGA timestamp counters at the same time when multiple ARCADIA-MD1 are connected to the same FPGA (the purpose of this routine is explained below);
- The sensor timestamp and the corresponding FPGA timestamp must be aligned. This operation is called timestamp synchronisation and is explained below; it is made possible by the presence of an offset to the FPGA timestamp that can be set via software.

On the other hand, messages given to the chip are used to (only the most important parameters are listed here):

- Slow down the sensor clock with respect to the input clock setting a multiplying factor;
- Tune some digital architecture parameters to optimise evacuation speed;
- Mask some signals (like clock, testpulses, and readout) in some sections;
- Enable space mode[1];
- Set the strength of the signals toward the FPGA;
- Configure some of the pixels in the array properly;
- Set some analogue architecture parameters to optimise the amplification or change the comparator threshold (at a section level).

3.2.3 The startup routine

When dealing with a new physical object, it is important to define a series of actions to perform whenever the device is powered up. These are meant to bring the chip to a well-known, fixed state. The startup actions are represented in Figure 3.9. Actions performed at startup are the following:

- **Connect FPGA:** instantiates the interface with the FPGA and verifies that the connection through the IPbus suite is working properly;
- **Enable Readout**: enables the readout of a set of sectors. By default, readout is enabled in all 16 sectors of the main demonstrator under study;
- **Enable Clock**: propagates the clock to all 16 sectors;
- **Enable Injection**: makes all pixels injectable;
- **Chip Initialise**: performs some checks on the ARCADIA chip to verify whether all sectors are working properly;
- **Stabilise Lanes**: checks that each sector is behaving as expected and masks those that are not working correctly;

[1] A mode capable of greatly reducing the power consumption by using only one serializer.

- **Set Timestamp Resolution**: sets the proper timestamp resolution in terms of number of clock cycles.

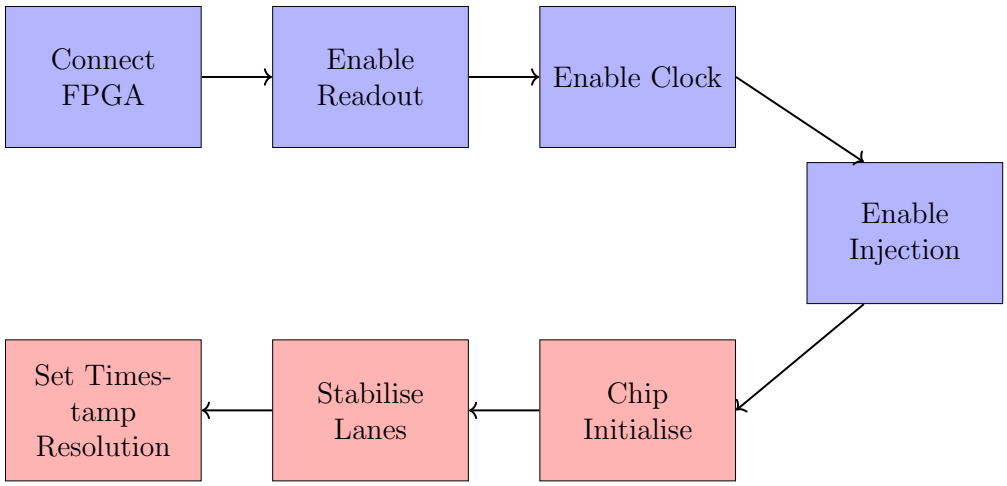


Figure 3.9: Actions implemented in arcadia-daq-sw startup. The red boxes show some complex tasks that are presented in detail below.

Chip Initialise

In Figure 3.10 **Chip Initialise** routine is opened in its components.

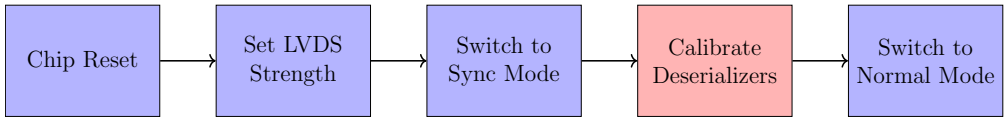


Figure 3.10: Actions implemented inside the Chip initialise method of the startup routine.

- **Chip Reset**: resets the device status, loading all the default configuration values for both the chip and the FPGA;
- **Set LVDS Strength**: sets the LVDS (**L**ow **V**oltage **D**ifferential **S**ignal - the protocol used for communication) output buffers' driving strength. By default, it is set to be the highest possible;
- **Switch to Sync Mode**: switches to synchronisation mode, where data output is not possible and serializers communicate only the idle word;
- **Calibrate Deserializers**: calibrates deserializers by looking for alignment with the idle word;
- **Switch to Normal Mode**: goes back to normal mode where serializers communicate data when available.

Deserializers calibration

The deserializers calibration task looks for the alignment in the communication with the chip: whenever in sync mode, each one of the 16 serializers gives a fixed output idle word. Depending on the time it takes for the word to reach the deserializer, this word may not be seen correctly due to setup or hold violations [107]. The FPGA lines used for this communication also host some delays (called taps) that can be added in order to change the phase of these signals on each of the serializer output. All possible taps are tested in this phase and the correct output word is sought. Whenever a plateau of correct words is found, the best tap is set, and it will be used through all the data taking phase. This way, data writing and sampling happen in well-separated moments in time. Sometimes, due to issues that arose during sensor production, some sectors do not respond correctly and do not show a clear plateau where the synchronisation word can be read successfully. As communication is not reliable, the readout of these sectors is deactivated.

Stabilise Lanes

Stabilise Lanes is used to group sectors depending on their behaviour. Each sector is tested and is thus recognised as:

- **Unsync** if, for every tap tested, the fixed sync word was not found and, therefore, the serializer is not reliable;
- **Noisy** if, when unmasked, data are read instantly even after a reset has just been given. This actually means that some data travel through the DAQ in the time it takes for the software to send two consecutive messages: the reset and the readout request. The research of noisy lanes can be deactivated if the detector is already in place, e.g., close to a radioactive source, but this operation is risky, as noisy lanes will not be spotted and masked properly;
- **Dead** if the sector is not responding to test pulses, although all the settings are correct;
- **Good** if it does not belong to any of the previous sets.

After this routine, only the good sectors remain readable, and all the others are disabled for the following operations.

Set timestamp resolution

Once the chip is in a known state, the timestamp resolution is set. This operation does not only set the correct registers choosing how often the timestamp counter is updated with respect to the clock, but also performs an operation called **timestamp synchronisation**. As introduced above, each data comes with two timestamps (see the first line in Figure 3.5): one coming from MD1 and another coming from the FPGA itself. They run with the same clock, but are misaligned. In the *set timestamp resolution* phase, a test pulse is sent to the array, and the two timestamps of the data item are checked. The difference between the FPGA timestamp and the MD1 timestamp is calculated and saved inside the FPGA as what

is called `timestamp_delta`. This difference is then accounted for every time new data is sent toward the computer: instead of giving the FPGA timestamp naked, delta is subtracted from it. This way, the last 8 bits of the FPGA timestamp will coincide with the MD1 timestamp. This operation is schematically depicted in Figure 3.11.

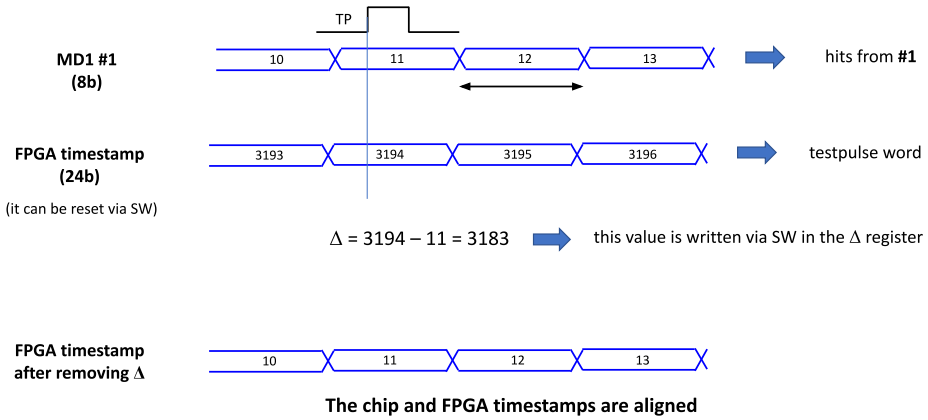


Figure 3.11: A representation of timestamp synchronisation between an ARCADIA-MD1 and FPGA. Courtesy of dr. D. Falchieri

Startup routine performance

The goal of the described startup routine is to establish a known state for the chip, study all the sectors, make good responding sectors injectable and unmasked, and deactivate bad responding lanes by masking and disabling the readout.

The performance of this routine can be discussed according to the software goals introduced in Section 3.2.1:

- **Completeness**: all preliminary tasks can be done to check the device functioning and change its configuration properly. Several tests are performed, both using test pulses and waiting for random hits due to very noisy pixels;
- **Easiness**: the startup routine, performed maintaining the default configuration can be done with one line of code. All the tasks are fragmented into minor tasks, all callable one-by-one to check separately each feature and solve each issue (all boxes in Figure 3.9 and Figure 3.10 are stand-alone methods that can be called one by one by a user willing to study in depth the chip performance);
- **Clarity**: all functions are commented, documented, and have self-explanatory names, making all the steps as clear as possible; many also come with a verbose mode to spot potential issues;
- **Portability**: the system, although quite portable between different Python, MatPlotLib and NumPy versions, currently works only in CentOS-8 Linux distribution, as there never was a push to make it more portable.

3.2.4 Data taking mode

During an experimental test aimed at the commissioning of a detector, after the startup routine is complete and successful, users are likely willing to carry out their own experiment by setting up a proper software environment. The most general programme of arcadia-daq-sw is called **data taking** and collects data, performs some statistics on the fly, prints them on screen, and saves them on disk. This routine allows for the preliminary setup of some parameters, making the users choose whether to enable or disable the single routines (see Table 3.2). This approach was shown to be useful because it allows for a good level of customisation, and a single easy-to-maintain software can be used for experiments in both low-rate environments (such as those involving cosmic rays or collimated sources) and high-rate environments. In this section, both the routines that must always be performed to take data and the ones that may be disabled depending on the environmental conditions are discussed.

Parameter	Meaning
`draw`	Enables real-time data drawing.
`Cluster`	Enables real time cluster analysis.
`blob_erase`	Enables blob erasing routine.
`trigger_mode`	Enables trigger mode.
`automask`	Enables auto-masking.
`vcasn`	An integer parameter to feed the chip as a threshold to use.

Table 3.2: Customisable parameters in data taking software. This is not the exhaustive list of parameters, but only those that enable or disable a routine.

Timestamp extension

The first fundamental task performed by this programme is the timestamp extension. As discussed previously, ARCADIA-MD1 provides data with its own 8 bits timestamp, and FPGA extends it to 24 bits. Taking into account that the timestamp counter period used as a reference is $\mathcal{O}(250\text{ ns})$[2], by using this setup the FPGA timestamp overflows in a few seconds. This means that in actual runs this timestamp will overflow several times. For this reason, the FPGA broadcasts a word, named *timestamp overflow* (it was shown in Figure 3.5) every time the counter overflows. By catching this word, it is possible to extend the timestamp arbitrarily, as the only limitation is the memory to use at a software level.

At the end, the timestamp kept as reference and saved to file is what is called the **extended timestamp**, which takes into account all the components that implement the

[2]A precise measurement of the time resolution has not been performed yet; its experimental estimation is one of the next tasks for the collaboration. The value reported here for the timestamp counter period is the one that was used experimentally and showed good synchronisation when a cosmic muon crosses two devices in a hodoscope experiment.

data timestamp. It is represented in Figure 3.12: the 8 LSBs are a copy of the MD1 timestamp, after that the 16 MSBs of the FPGA timestamp are used (reaching a total of 24 bits); then the software timestamp (arbitrarily long) is added. MD1 timestamp is used, although it should be the same as FPGA timestamp, because it is more precise. In fact, if the particle rate is particularly high, data may be queued in the chip periphery, waiting some clock cycles for their evacuation. As a result, the arrival of the data into the FPGA can be delayed. The chip timestamp, on the other hand, is associated to the hit address as soon as the information reaches the periphery, before the queuing toward the serializer, and is thus the most precise available information.

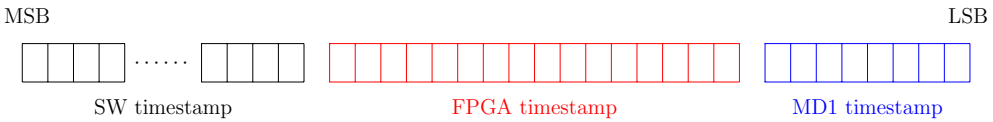


Figure 3.12: Extended timestamp. Its length is arbitrary, depending on the memory allocated at software level.

In order for the extended timestamp structure to work as expected, it is important not to lose the timestamp overflow word. This might happen because the FPGA memory is full, in this case the actual timing information is biased, and it may be possible to see as synchronised data that really are one FPGA timestamp full cycle distant. In order to solve this issue, in the future a firmware fix consisting in the impossibility to fill the FIFO with data is envisioned; such as there would always be some space in the FPGA memory for these overflow words.

Visualisation

When visualisation is enabled, the user can see two different plots on the same window that can be used to understand how the experiment is going: on one side the integrated data from the beginning of the run, on the other side a plot showing the last set of data slowly fading (a data fades in around 10 s). These plots are called **maps** through this chapter, as they are a map of the pixels that collected the data. In the next section, some maps containing all the data taken within an experimental run are shown and commented on.

Data Saving

During data acquisition, data are saved on disk every time they are read in the software. They are saved in CSV (**C**omma **S**eparated **V**alue) format, and they contain every information for further offline analysis. Each saved data contains:

- The address of the pixel region (in terms of section address, double column address, and pixel region address);
- The hitmap of the pixel region;
- All the timestamps (MD1, FPGA, software, extended one);

- Eventual increasing trigger number (how many triggers arrived in the FPGA before this readout).

Clustering

In clustering mode, whenever a new hit reaches the computer, close ones are sought. Different hits are added to the same cluster one by one if the lowest Chebyschev distance between the pixel and one belogning to the cluster is one (which means that they touch, at least, at a single point, as can be seen in Figure 3.13), and if they are within the same (predefined) timestamp window. When clustering is enabled, some statistical properties of clusters are calculated and plotted online, such as the timing distribution of the clusters (hence the rate), their size, and their asymmetry. Although quite interesting, the use of the clustering algorithm is quite tricky: the computational power necessary to cycle over the hits to look for clusters is relatively high, and clustering can be successfully used only with low rates (e.g., cosmic rays or weak sources). This clustering algorithm is performed at a software level and is executed after the data have been read and are currently on disk. It has nothing to do with the clustering readout implemented at the silicon level of ARCADIA-MD1.

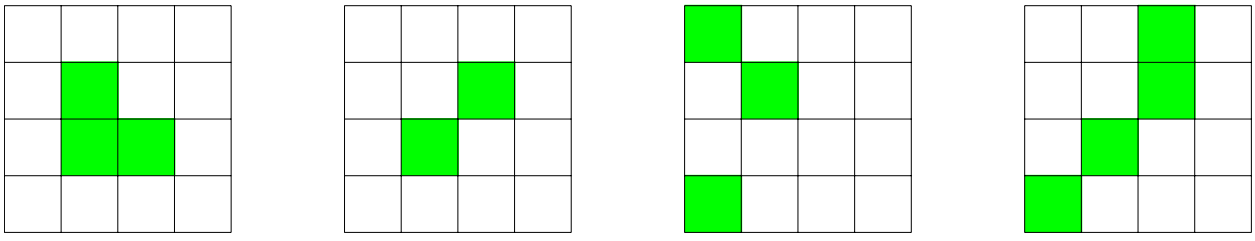


Figure 3.13: Some examples of how clustering algorithm works.

Blob erasing

Due to some issues in the very first experimental campaign run with ARCADIA-MD1, often *blobs* were spotted. Blobs are defined as round-shaped clusters with a diameter greater than 15 pixels (therefore quite large: considering that the pitch of ARCADIA-MD1 is 25 µm, these blobs have a diameter greater than 300 µm). The origin of this phenomenon was not explained, since by changing the operating point (the bias voltage) it became no more statistically relevant (a blob can still be seen every $\approx$ hour on a $12.8 \times 12.8\,\mathrm{mm}^2$ sensor). The presence of these objects may be caused by setting too low a voltage on the sensor: a weak electric field, insufficient to collect charge in the nearest well in a device starting the depletion from the bottom, may allow charge to diffuse from a hit pixel to adjacent ones, thereby creating the impression of a circular shape of hits. If this effect were more relevant, a more detailed diagnosis would have been made. To allow efficient data taking also in this problematic operating point, arcadia-daq-sw can produce a second map containing the overall hits since the start of the acquisition that does not show these blobs. This feature

can be used only when clustering is already active, and is similarly quite computational demanding.

Triggering

In the setup discussed previously, the trigger can be propagated only at the FPGA level, as ARCADIA-MD1 itself can work only in data push mode; i.e., data are propagated to the FPGA every time they are available, independently of any external signal. When trigger mode is enabled, data saving in the FPGA memory is allowed only when an internal signal (called `trigger_enable`) is high. Whenever an external trigger signal reaches the FPGA, `trigger_enable` is kept high for a customisable number of clock cycles. Only data coming from the chip in this timing window are successfully written to the FPGA memory and, later, read by the computer.

A future upgrade of the trigger could include the possibility to work with *late triggers*; i.e., triggers arriving after the data information. This was not a priority in the early commissioning phase, as the tests were performed using as a trigger a scintillating tile read via a SiPM, faster than ARCADIA-MD1 in terms of data transmission.

Automask

When automask is active, every fixed (and customisable) period of time, statistics are calculated over populated pixels: all pixels that received at least one hit are considered, and the average and root mean square of the hits since the beginning of the run are calculated. Then, if a pixel has a hit count h:

$$h - \bar{h} \geq A_s \cdot \mathrm{RMS}$$

With A_s the `automask_s` a parameter setting the automask strength, then this pixel is masked and will not receive any more values, as it is considered a noisy pixel. This mode helps to interpret and visualise the data.

Automask can work in two different regimes, either via hardware or software:

- in **hardware automask** the noisy pixel is physically masked by configuring the sensor itself. This operation takes some time and might be problematic if the chip throughput is too high, but the chip throughput is reduced after the successful masking;
- in **software automask** the noisy pixel is no longer represented in the online plots that are used to check the status of the ongoing acquisition, but physically its data continue to flow toward the computer and will be saved on disk. This can be useful if acquisition speed is crucial.

This routine is necessary because, just like in several other silicon pixel sensors, there is a set of pixels that are inherently noisy because of issues during production. The addresses of such pixels remain fixed through the different experimental runs performed. On the sensors discussed here, the fraction of noisy pixels found experimentally is something of the order

of 1 per mille (without considering the digital electronics issues resulting in full columns or sectors malfunctioning).

Data taking software performance

- **Completeness**: the software actually takes data and implements some basic statistical analysis. There is a delicate trade-off between the push to have several useful plots and analysis, and the need to run everything online. The overall performance might be improved, optimising the software speed and adding more tools, but the software is anyhow quite flexible and fit for the device commissioning;
- **Easiness**: the programme runs by default, setting up just some parameters on the top. The modification of some plots and the addition of other studies remain a prerogative of the developers and not of the users; this will be improved in the future;
- **Clarity**: there are several comments throughout the code that help the user understand why some lines are important. On this side there may be an improvement as well;
- **Portability**: just like the startup routine, neither this code has ever been ported to a system different than CentOS Stream 8.

Experimental results

Several experimental campaigns were run by members of the ARCADIA collaboration using the software described above. In Figure 3.14 it is possible to see an example of a recent output taken in a 5 day long cosmic ray run using the data taking programme.

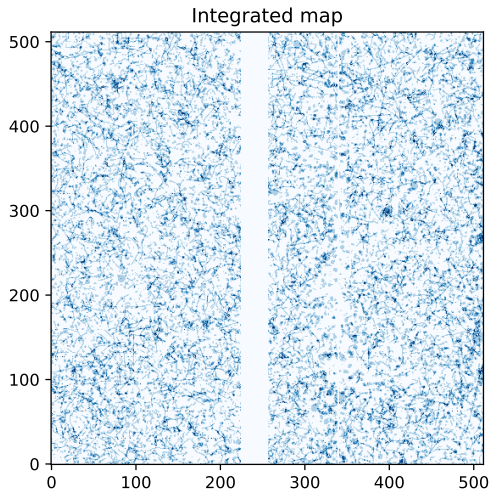
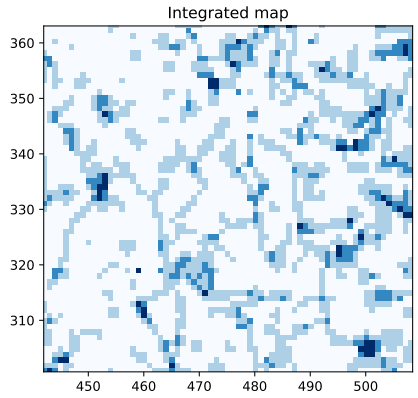


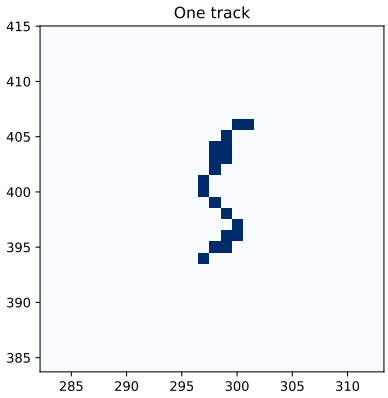
Figure 3.14: The cosmic muon tracks through one device accumulated over 5 days of data taking. White column due to hardware issues during the sensor production.

A complete analysis is not reported here, but some minor adjustments were made to

produce this image. In particular, the automask routine was used to mask noisy pixels. In this sensor, all sectors are good, hence they all get activated by the startup routine; nevertheless, for this particular sensor, section 7 had always to be manually masked, as it is far more noisy than the rest of the array; another column (339, hardly seen) was completely masked for the same reason. Moreover, there are still some noisy pixels (not spot by the automasking routine) probably due to problems in the digital electronics. Their value is reset offline. From a figure like this (one of many), a zoomed image can be extracted (Figure 3.15a) and a particular cluster can be isolated (Figure 3.15b). A few clusters are symmetric, but most are filamentary and track-like. The former are probably due to muons traversing the chip; the latter are probably due to muon-produced δ-rays travelling inside the array itself. This interpretation was tested by increasing the threshold value of the detector: the number of filamentary clusters decreases leaving more symmetrical and less extensive ones; nevertheless, in order to be sure about this interpretation, much more data taking is necessary, as well as a thorough statistical analysis, and Monte Carlo simulations.



(a) A zoom of Figure 3.14. There are few relatively symmetric clusters and numerous filamentary ones.



(b) A single filamentary cluster in an ARCADIA chip. The shape of this track-like cluster is typical of δ-rays produced by traversing muons.

Figure 3.15: Some images taken by using arcadia-daq-sw with an ARCADIA-MD1 and cosmic muons.

3.2.5 Multi-sensor mode

By using more than one sensor, it is possible to perform several tests to evaluate the performance of ARCADIA-MD1 (such as calculating sensor efficiency); hence, a dedicated programme was developed. The results presented in this section are preliminary, as the experimental campaign is still ongoing. The tests carried out so far were done by creating a sensors **hodoscope**; i.e., a telescope of different ARCADIA-MD1 sensors (up to three) stacked one above the other in parallel. In order to read from multiple sensors, both preliminary operations must be adapted accordingly, and the data taking software must be adjusted, as some operations need to be rethought. In this section, the modifications performed to data taking

software to allow the readout of different sensors are discussed. When dealing with more than one sensor, a formal definition of **event**; i.e., a single particle creating data in more than one tracking plane, is necessary. Here, an event is defined as "a set of hits in at least two detectors within a fixed timing window" (usually 2 timestamp periods). This definition of event relies only on the temporal proximity of the hits, not on the spatial one.

Software loop

When there is only one connected device, a separate thread can be used to read from the ethernet line: data are continuously read and whenever there is new information, it is digested by the software, plotting the hit pixel and updating the saved data. If, on the other hand, multiple devices are connected to the same FPGA, readout must be queued, as only one device can transmit through the cable at a time. Hence, the programme is implemented in four main groups, continuously looped: First, new data are read from detector 0 (plots and files on disk are updated); then the same happens for detector 1; then for detector 2 and, finally, once all detectors have been read, there is a loop seeking for coincidences between at least two different devices. After a full software cycle, the plots related to events (described later) are updated.

This modification leads to a worse-performing structure, as the readout now is performed in the very same thread responsible for the analysis. An upgrade consisting in using separate thread for the analysis and for the alternated readout of the sensors is planned in the future versions of this programme.

Preliminary operations

When multiple sensors are connected to the same FPGA, they operate completely independently; therefore, all the startup operations must be repeated once for every device. Nevertheless, after these operations, the sensors are non synchronised as their timestamps, although fed with the same clock, run on different tracks. To sync the different devices, another synchronisation routine must be performed. This is performed by using the FPGA reset timestamp signal, broadcast through software, and shared between all the timestamp counters. After the common preliminary phase, when all the chips are studied singularly and bad-responding sectors are disabled on each one, this command is broadcast, resetting all the FPGA timestamp counters at the same time. With this operation, the three counters start running synchronised, but the equivalence between MD1 timestamp and the last 8 bits of FPGA timestamp represented in Figure 3.11 is lost. In current implementation, losing a little bit of timing precision, when constructing the extended timestamp the MD1 timestamp is neglected and only the FPGA timestamp is used (hence also Figure 3.12 is not valid anymore, as all the 24 LSBs of the extended timestamp come from the FPGA). In Figure 3.16 it is possible to see a scheme representing the synchronisation of 3 devices.

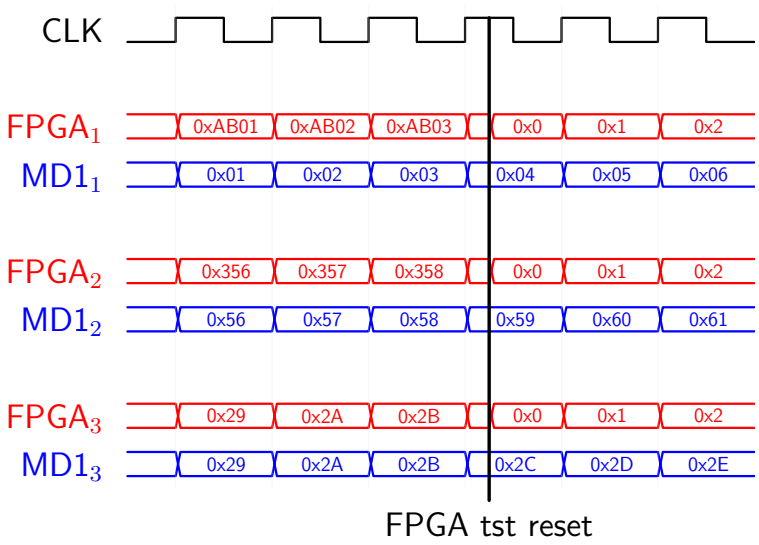


Figure 3.16: Synchronisation of three different ARCADIA-MD1 mounted on the same FPGA. On the left side it is possible to see that the FPGA timestamp is just an extension of the MD1 timestamp (the last 8 bits are the same). After the synchronisation operation, this equivalence is lost, but the three FPGA timestamps are correctly synced. The jitter between the various counters has not been drawn for clarity.

Coincidences seeker

To look for coincidences without looping every time through the entire set of data, a smart coincidence seeker is executed online. It starts from the three sets of data (collected so far), one for each detector, and its purpose is to find the events; i.e., the sets of data for which there is a hit on at least two devices in a timing window. Some details on the implemented code can be found in Appendix B.

The advantage of this algorithm is that it does not need to run every time from the beginning of the data sets. After it is run for a second time, it starts looking for coincidences from the last data of the previous readout. This allows the algorithm to run online, during data acquisition, as its execution time depends only on the size of the last readout, not on all the data read so far. The limit of this algorithm lies in the fact that it works only if the data are ordered. This, although not being in principle granted, was found to be true experimentally when using cosmic rays or collimated radioactive sources.

Verification plots

When running in multi-device mode, several plots are automatically activated in order to check the correct functioning of the devices:

- A **synced figure** represents on three different bidimensional scatter plots the events data; i.e., the hitmap of the data only in the cases when there was a hit on another detector as well (see Figure 3.18);

- A **spatial correlation figure** represents on two one-dimensional histograms the difference between the average X and average Y of a random hit inside the event for 2 random detectors. If the hodoscope is aligned and the source is collimated (i.e., the ionising radiation is perpendicular to the chips), there should be a peak at zero on both dimensions;
- A **timing correlation figure** with a one-dimensional histogram plotting the difference between the timestamps that originated an event. If synchronisation routine is working as expected, there should be a peak around zero; on the other hand, if chips were independent a uniform-like figure due to random correlations between different particles should be seen (see Figure 3.17).

These plots are a helpful and quick way to make online checks to see if the acquisition makes sense or not; they do not consume too much computation time, which can be still devoted to the continuous readout of data and their saving on disk. By looking at the whole data set, it is then possible to perform more precise and significant statistical analysis offline.

Experimental results

An experimental campaign was run to check whether the software works as expected. The first meaningful result can be seen by looking at the timing correlation histogram (Figure 3.17): in this case, a cosmic run was performed using 2 detectors and a timing window of 200 timestamp units (corresponding to 50 µs).

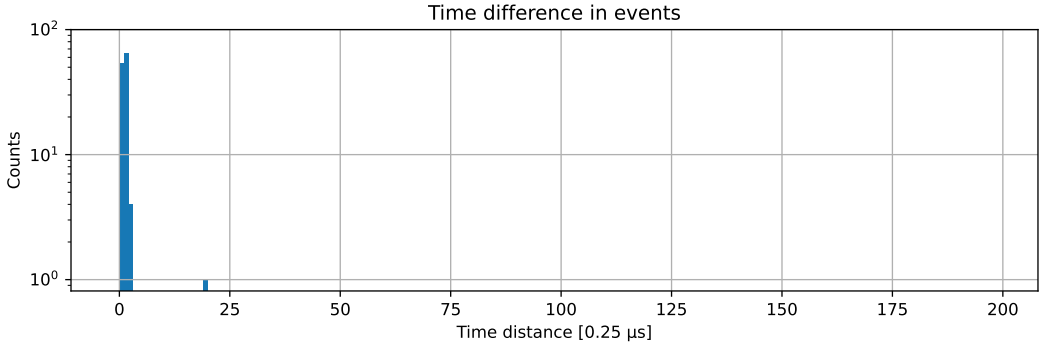
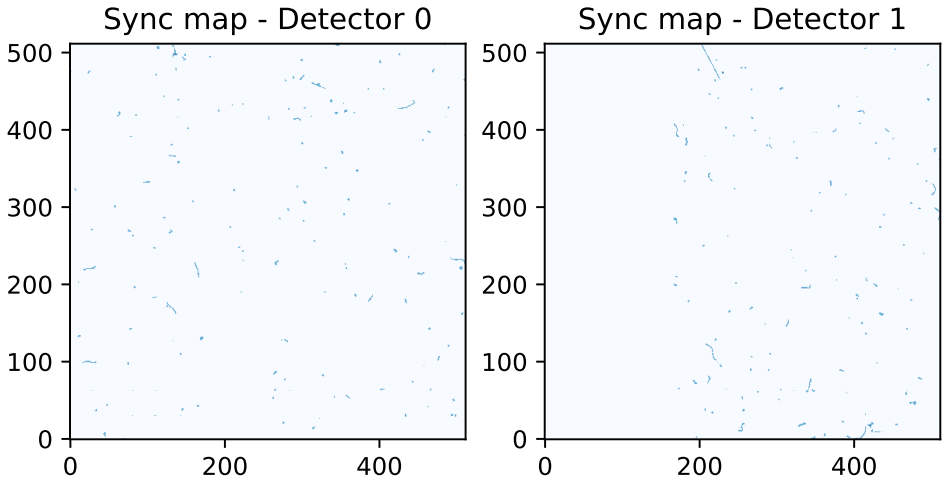


Figure 3.17: The online histogram showing time difference between two random hits belonging to the same event. In this case, two detectors were used, and the event definition was constrained to time difference of 200 timestamp units (50 µs). Logarithmic y scale.

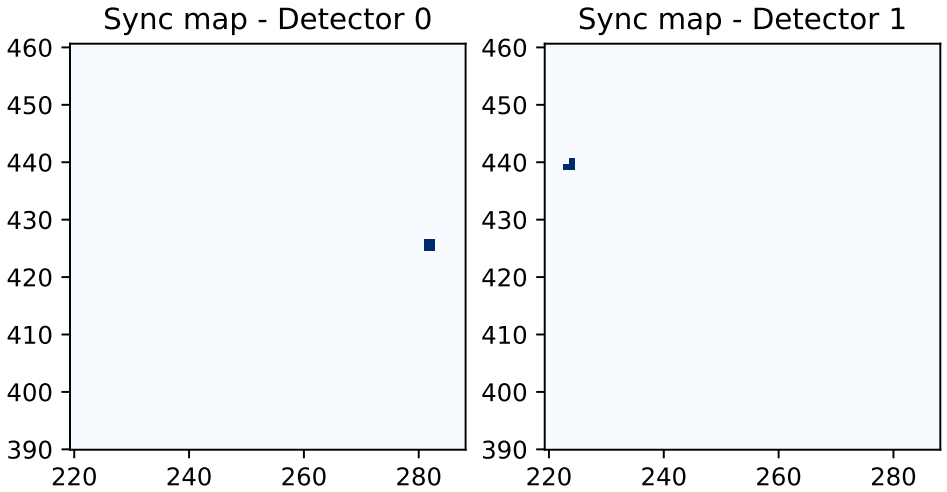
This histogram clearly shows how the synchronisation works as expected: the peak at zero is quite glaring. If synchronisation was not correct, there would be a peak somewhere else in this histogram, while if synchronisation was not performed at all the histogram would resemble a uniform distribution of data, representing random coincidences.

In Figure 3.18a it is possible to see the hitmap of the two detectors only when there is a hit on both of them. From this image it is notable that most of these clusters are larger than one hit only. This suggests that these clusters are related to real particles and not

electronic noise, which is most likely to happen on a single pixel at the time. As a last check, Figure 3.18b shows an event consisting of a single comic muon crossing the detector 0 (the upper one) and 1, almost perpendicular to the detector planes. Also in this case more than one pixel was high in the hitmap, highlighting how unlikely this event is to be caused by electronic noise.



(a) The two hitmaps of the detector whenever there is at least one hit on both of them. Detector 0 has a dead sector (number 7), while in detector 1 the five leftmost sectors are not working properly, as this detector had a few production issues.



(b) An event consisting of a muon crossing almost perpendicularly the two tracking planes. This figure was obtained by selecting one event and zooming Figure 3.18a.

Figure 3.18: Some plots showing data acquired using a hodoscope with 2 devices.

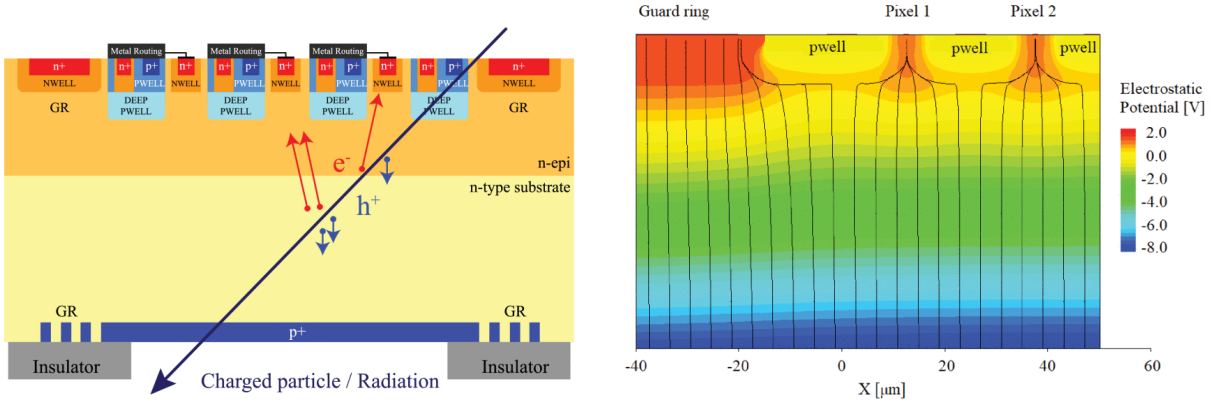
3.3 Bulk damage experiment using neutrons

I contributed, within the ARCADIA collaboration, to an experiment on the sensor resistance to bulk damage by using neutrons. Some introductory description of radiation damage, as well as an introduction to its modelling, can be found in the Appendix A; this section focuses on the experimental campaign and the results obtained, first presenting the methodology in Section 3.3.1; then a subset of the results obtained, favouring the most significant ones, in Section 3.3.2.

3.3.1 Instruments and Methods

Sensor used

The MATISSE sensor is developed in the very same modified 110 nm LFoundry CMOS technological process used for the ARCADIA sensor [30] (indeed it is considered a precursor of the ARCADIA sensor). This experimental campaign was carried out within the framework of ARCADIA development, as a demonstration that MATISSE can successfully resist displacement damage is a good sign in hoping that devices developed in the same technology (hence ARCADIA and, for the purposes of this thesis, even a device hosting the hash architecture discussed in Chapter 4) may behave similarly. Just like ARCADIA, MATISSE uses backside processing to create a junction that can be biased to deplete the whole sensor substrate. It is described in detail in [72]. The cross section of the sensor can be seen in Figure 3.19a. The deep p-well is used to prevent the n-wells hosting the p-MOSFETs from collecting the charges generated by the impinging charged particles, and the n-doped epitaxial layer is used to limit the punch-through current between the back-side and the deep p-well when the sensor is fully depleted. A TCAD simulated potential profile of two pixels can be seen in Figure 3.19b.



(a) A scheme of the cross section of MATISSE sensor.

(b) Simulated 2-D potential profile and electric-field lines at full depletion in MATISSE sensor. Only the sensor surface region is shown.

Figure 3.19: MATISSE sensor cross section scheme and simulation. Taken from [72].

The test structures used implement a 576-active pixel array organised in four sectors, each consisting of 6 columns and 24 rows. Each sector implements a slightly different geometry for the collection diode, hence the pixels performance changes from sector to sector. At the end of the array, an end-of-column block manages the whole logic, handles configuration and controls the data transmission (the readout logic is the one described in [108]) . In a typical case, the whole array can be read in less than 30 µs using a 5 MHz clock. The pixel pitch is 50 µm and the sensing electrode, with its surrounding area clear of electronic circuits, occupies a region of $20 \times 20\,\mu m^2$ centred in the area of the pixels. The readout is performed in correlated double sampling; i.e., it is possible to read both the baseline and the signal. The post-layout simulation for the pixel gain resulted in $130\,mV\,fC^{-1}$, hence a MIP should create in 300 µm around 500 mV. A list of the test structures used for this experiment can be seen in Table 3.3.

Sensor number	Irradiation Fluence [1 MeV n_{eq}/cm^2]
B10	0
B2	1×10^{12}
B9	1×10^{12}
B12	1×10^{13}
B15	1×10^{13}
B4	1×10^{14}
B5	1×10^{14}

Table 3.3: MATISSE sensors used for the bulk damage study with their irradiation level.

Irradiation

MATISSE sensors were irradiated using neutrons as damaging particles in a research reactor. Neutrons, compared to charged hadrons, have an effect that can be easier to normalise; they are a source of **N**on-**I**onising **E**nergy **L**oss (NIEL) but not (at least directly) of **T**otal **I**onisation **D**ose (TID) (see Appendix A). In the reactor, a background of ionising gamma rays was present, but the total deposited ionisation dose was known for every irradiated sample.

The reactor that was used is the TRIGA Mark II research reactor in Ljubljana [109], [110]. This reactor provides a uniform fast neutron flux, tunable from 2×10^9 to 2×10^{12} $n/cm^2\,s$. Moreover, it can work in pulse mode operation obtaining 1×10^{14} n/cm^2 in 20 ms. The neutron spectrum can be seen (measured and simulated) in Figure 3.20.

For the irradiation, the devices were first glued and bonded to their mezzanines, then characterised in the laboratory and sent to the reactor, where they were irradiated. The

damage to the electronics on the mezzanine led to some technical issues, discussed below.

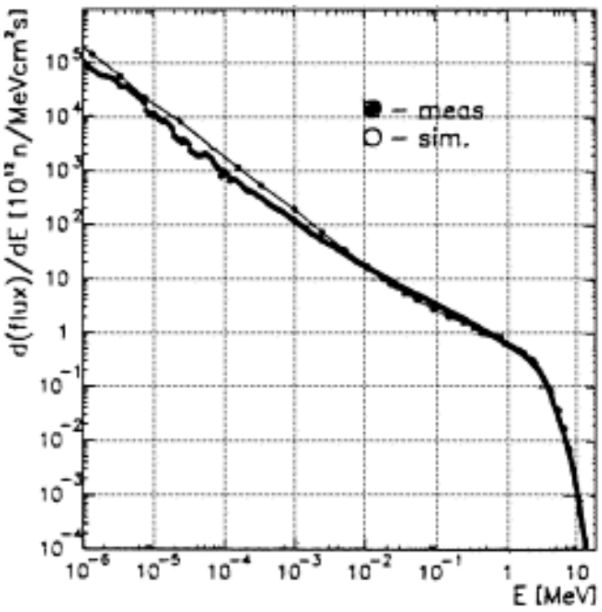


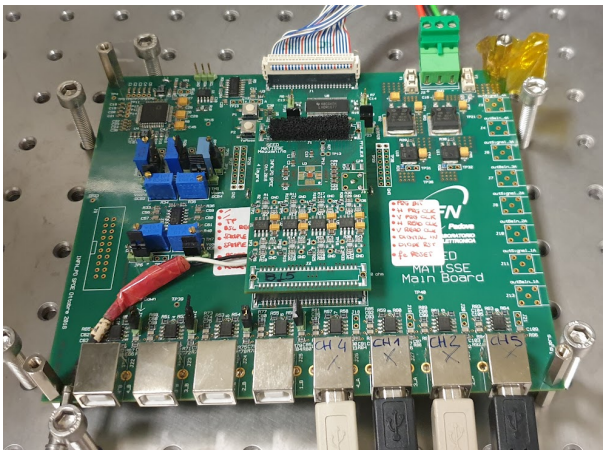
Figure 3.20: The measured and simulated neutron spectra of the TRIGA irradiation facility. Taken From [110].

Measurement

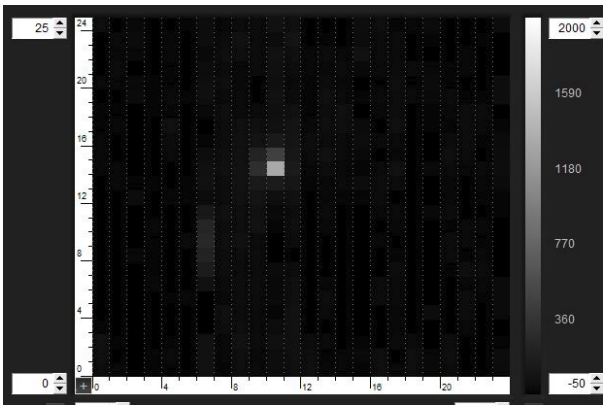
Since the irradiation, the devices were kept frozen at about −10 °C to limit the annealing of the defects produced. The thermal history of every single device was recorded to reconstruct the effects of the unwanted room-temperature annealing cycles necessary to measure the chip performance. For operational measurements, an X-ray source of ^{55}Fe, an isotope that decays only by electron capture to ^{55}Mn, thereby emitting a quasi-monoenergetic X-ray of 5.9 keV[3] was used. For such an X-ray in silicon, the attenuation constant is $1.470 \times 10^2\,\mathrm{cm^2\,g^{-1}}$ [112], which translates into an absorption coefficient of ≈250 nm, that means that most of the energy is released in the first micrometer of silicon.

The radioactive source was placed on the sensor, as close as possible without damaging the wire bondings (≈ 2 cm) and analogue data were acquired using the DAQ that was already developed by the SEED collaboration that created MATISSE. The entire setup was placed in a cooling box with controllable temperature. This study was performed on the available chips varying both the setup temperature and the bias voltage. A photo of one of the sensors mounted on its readout board can be seen in Figure 3.21a, while in Figure 3.21b a real-time acquisition screen shot is shown.

[3]Actually, there are various channels available. Neglecting the most unlikely X-ray emission channels, in about 16% of cases an X-ray with energy 5.898 75 keV is released, and in about 8% of cases an X-ray of energy 5.887 65 keV is emitted. In 60% of the cases, energy is given to Auger electrons [111].



(a) A photo of the sensor mounted on its readout board, ready for measurement.



(b) A screenshot of the data analysis software. Here a cluster due to an impinging particle is visible.

Figure 3.21: Measurements on irradiated sensors; setup mounted in Padova.

Protocol

The measurements protocol adopted was:

- Starting from the lowest temperature used (around $-10\,°C$) the device was biased at the highest achievable voltage. Before irradiation, it was possible to bias these devices up to 250 V, far above the threshold for full depletion. After irradiation, this became impossible for some devices, as the leakage current was too high. First, the highest voltage that kept the leakage current under control[4] was set;
- When the bias voltage was set, the noise[5] was measured as well as the baseline for every pixel in the array;
- After this preliminary evaluation, the signal generated by the ^{55}Fe X-rays was measured, and a spectrum was produced. These data were used to compute the physical variables described later in this section; ultimately leading to the calibration of the system;
- These steps were repeated by first changing the voltage and then the temperature.

3.3.2 Results

Data definition

The MATISSE device, differently from both ARCADIA-MD1 and ALPIDE, is an **analogue device**; hence, it measures also the energy deposit and data contain both the position of the pixels and the amount of charge collected. Moreover, as previously described, the sensor

[4] All highest values for the analysed sensors lie between 130 V and 250 V.

[5] Throughout this section noise refers to the standard deviation of repeated measurements of the baseline, calculated experimentally.

acquires in correlated double sampling: i.e., the height of the voltage signal already takes into account the baseline voltage. To define when a signal can be considered a good **seed** for a cluster due to the passage of an ionising particle, the signal height is compared with the corresponding pixel noise, that was measured for every pixel. To run the experiment, the following parameters had to be set:

- **seed threshold**: How high with respect to the baseline a signal has to be in order to be deemed interesting and not be rejected as a mere statistical fluctuation. This threshold was set to be 6 times the noise;
- **matrix size**: When a seed is found, a square region, centred on the seed, is selected and the energy deposited in all pixels in such selected matrix is saved. The sum of the signals in these pixels is called the **matrix energy** value. The matrix size was set to 5; i.e., every time a seed is found, the voltage of a square group of 25 pixels centred in the seed is saved;
- **cluster threshold**: When a seed is found, there is a research in the aforementioned matrix for other pixels having energy higher than a given threshold[6]. Then, the output of these pixels is summed forming the **cluster energy**. The cluster threshold was set to be 4 times the noise.

The following quantities have been calculated from the experimental data:

- Seed energy, matrix energy, cluster energy. These quantities are calculated as one per detected seed;
- Multiplicity, defined as the number of pixels in the array above the cluster threshold. Also this quantity is calculated once per detected seed;
- Single pixel clusters energy, defined as the energy of the events with multiplicity one. This quantity is calculated only in single-pixel clusters;
- Noise. Each pixel has different noise; therefore, considering the size of the MATISSE, there are 576 entries.

Considering that every sector is physically different, all these quantities have been calculated for a single sector. For some quantities (such as the matrix energy, the cluster energy and the multiplicity), only the central part of the sector was used to avoid including data from the neighbouring sector in all the matrix-related calculations.

Moreover, some maps are presented showing the baseline and the noise values for every pixel in the array; these will give some hints on the correct functioning of the whole device.

Non-irradiated sensor

Figure 3.22 shows the two maps of the baseline and the noise of the the non-irradiated sensor B10. The conditions for this measurement were $-8\,^{\circ}\mathrm{C}$ and 130 V bias voltage. All sectors appear to be working properly and the device appears rather uniform, there are only a few pixels not responding as expected; this is due to production defects.

[6]This threshold is different from the seed threshold and must be strictly not greater than the seed threshold to make sense.

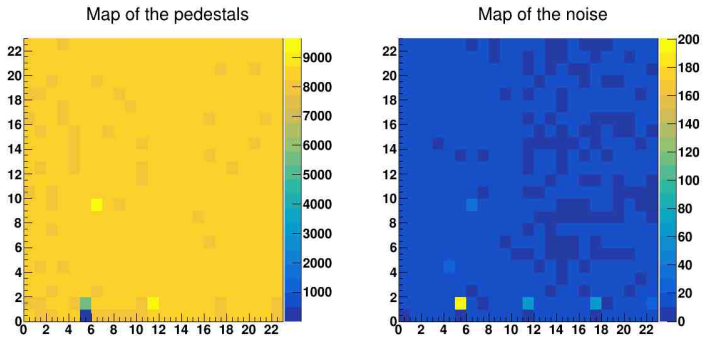


Figure 3.22: The baseline (on the left) and noise map (on the right) of the B10 non-irradiated sensor. $V = 130\,\mathrm{V}$; $T = -8\,^{\circ}\mathrm{C}$. All sectors are working properly, and there are only few pixels not responding as expected.

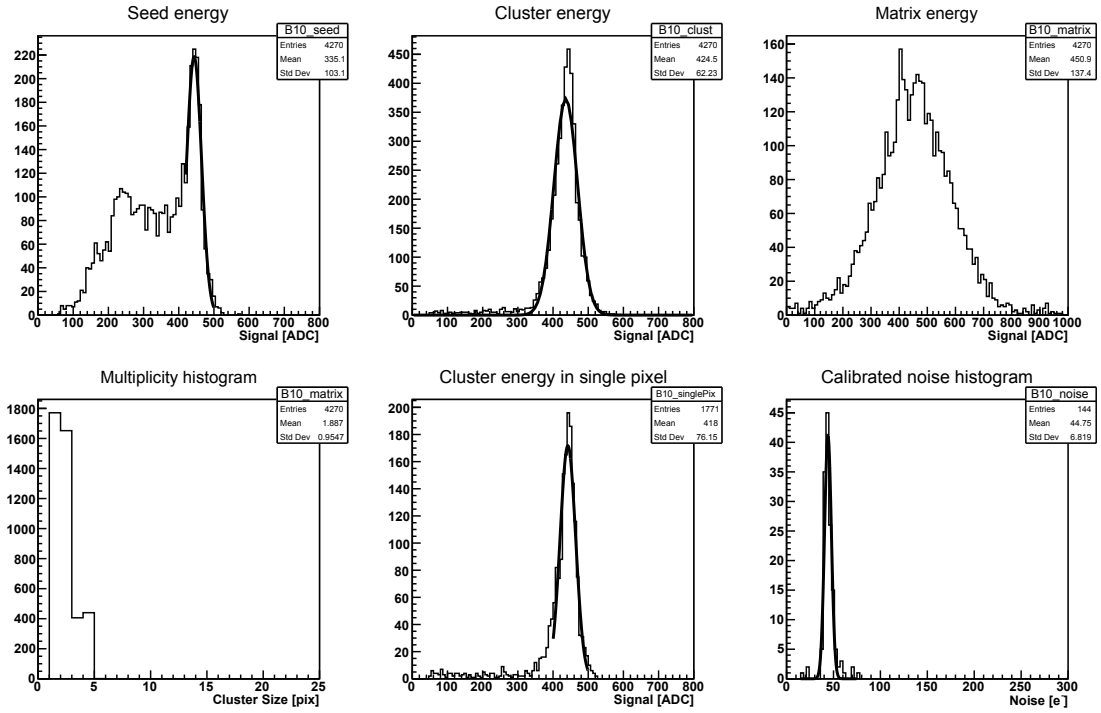


Figure 3.23: The performances of sector 1 in non-irradiated B10 sensor. $V = 130\,\mathrm{V}$; $T = -8\,^{\circ}\mathrm{C}$. Noise calibration performed by using the cluster energy. Fit line in the seed energy plot (top left) is used to confirm the value obtained in the single energy pixel plot (bottom middle).

Figure 3.23, on the other hand, shows the variables measured during data acquisition. These plots were obtained studying the leftmost sector; i.e., sector 1 (that was found the best performing in all the devices studied). The sensor and experimental conditions were the same. 4270 events were measured (hence the entries in the first 4 histograms); the bottom middle one is populated only with events that have a multiplicity value equal to one (in this case, 1771), and the rightmost at the bottom represents the noise of every pixel, hence is populated once per pixel (there are 144 pixels in a sector).

Irradiated sensor

For comparison, in Figure 3.24 and Figure 3.25, there are the same plots taken under the same conditions with the irradiated sensor B12, damaged with 1×10^{13} 1 MeV n_{eq}/cm^2.

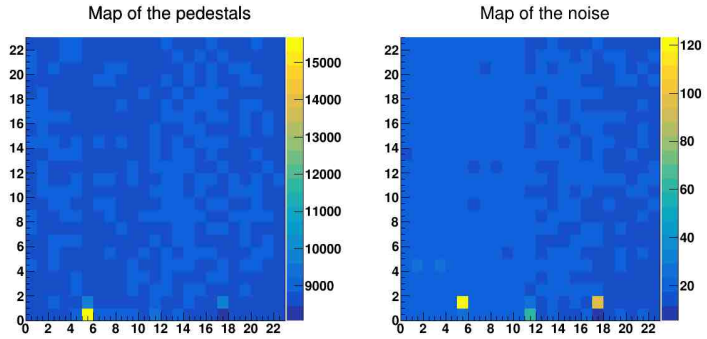


Figure 3.24: The baseline (on the left) and noise map (on the right) on the B12 sensor, irradiated with a fluence of 1×10^{13} 1 MeV n_{eq}/cm^2. $V = 130$ V; $T = -8$ °C. Still some pixels not working properly.

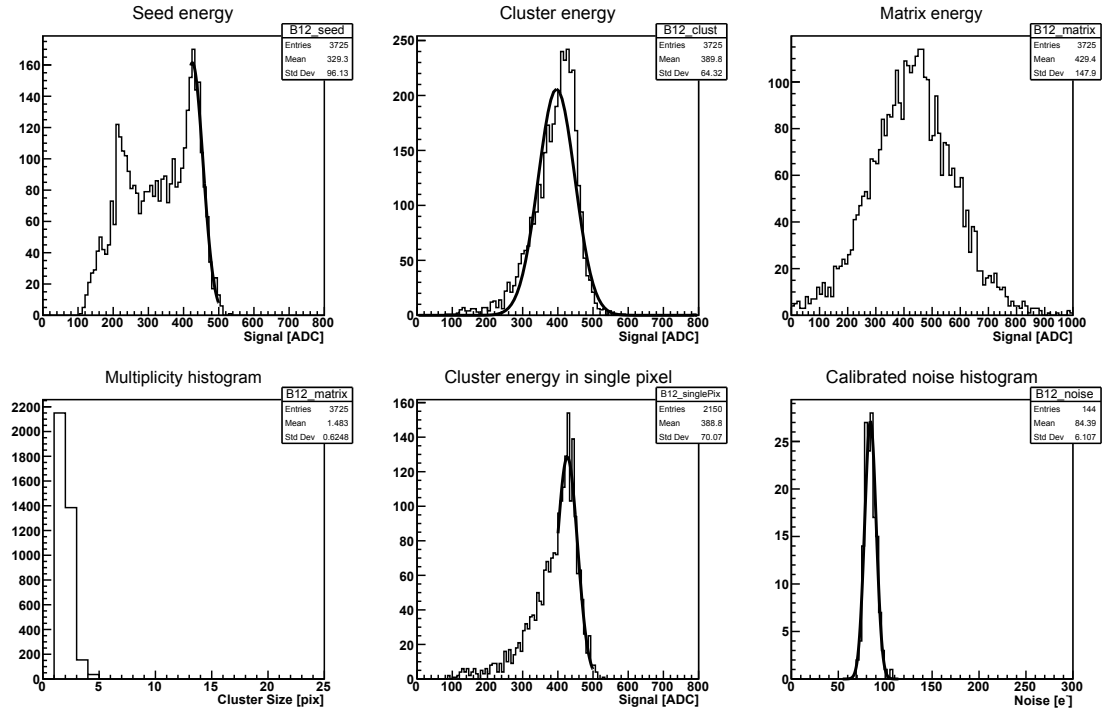


Figure 3.25: The performances of sector 1 in B12 sensor, irradiated with a fluence of 1×10^{13} 1 MeV n_{eq}/cm^2. $V = 130$ V; $T = -8$ °C. Same analysis routine used for Figure 3.23.

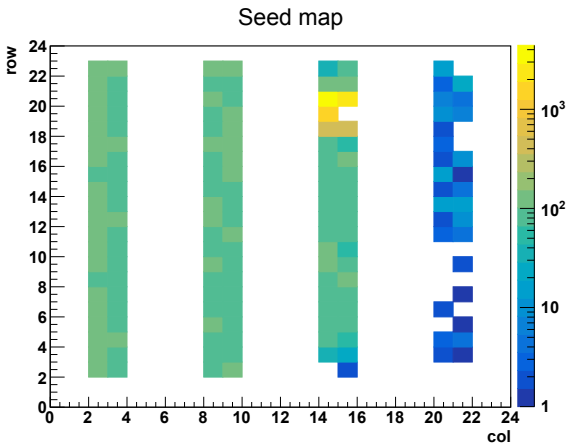
By comparing these figures, it is possible to see that:

- gain is similar: there is no statistical evidence in the change of the **C**harge **C**ollection **E**fficiency (CCE, see Section A.2). This suggests that full depletion is reached with the same voltage also in this irradiated sensor;

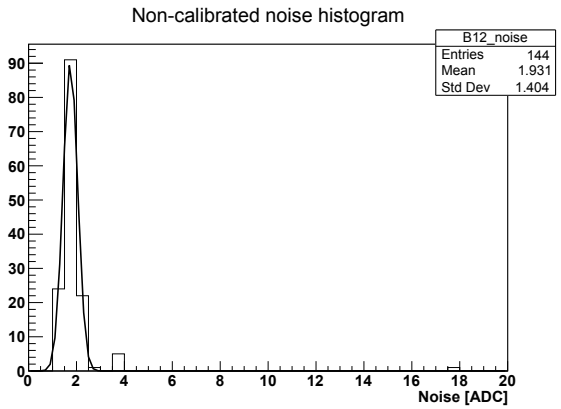
- both the seed energy and the single pixel cluster energy show a clear long tail on the low-energy side of the distribution in the case of the irradiated sensor. This might be due to the presence of charge sharing among neighbouring pixels: the bulk damage may have changed the effective doping, thus the shape of the electric field in the sensor. This effect was predicted by the sensor design group using Monte Carlo simulations;
- noise is significantly higher in the case of the irradiated device (around a two-fold increase);
- multiplicity is lower for the irradiated sensor. This might be an effect of the higher noise, as the signal threshold is set with respect to the noise.

Irradiated sensors issues

In this comparison, well-responding detectors were shown; during the measurements, some problems have been found. For example, as some pixel geometry designs were bolder than others, not all sectors were functioning properly in certain devices for the reachable bias voltages. Some were not responding properly even before the irradiation campaign. This is the case, for example, of sensor B2, where the bias voltage could not be pushed above 130 V and, at that value, the two rightmost sectors (3 and 4) were not depleted yet. This is shown in Figure 3.26a, where the map of the seeds position is reported. The white space dividing the sector is linked to the fact that for this analysis data with seeds close to the section boundaries were not included, as matrices overflow to the neighbouring section.



(a) The map of seeds (pixels with highest signal taken as cluster center) in B2 sensor (fluence 1×10^{12} 1 MeV n_{eq}/cm^2). The scale is logarithmic. the third sector has some extremely noisy pixels, and the rightmost columns do not give signal as it is probably not yet depleted.



(b) The extremely low noise histogram for sector 1 in B12 sensor (fluence 1×10^{13} 1 MeV n_{eq}/cm^2) at 23 °C. Plot not calibrated as it was not possible to perform full statistical analysis. Working sensors usually show a far higher average noise, around 10 ADC counts. Similar results were found in the other sensors with the same radiation level at the same temperature, independently from the bias voltage set. This result can be explained by a very high leakage current.

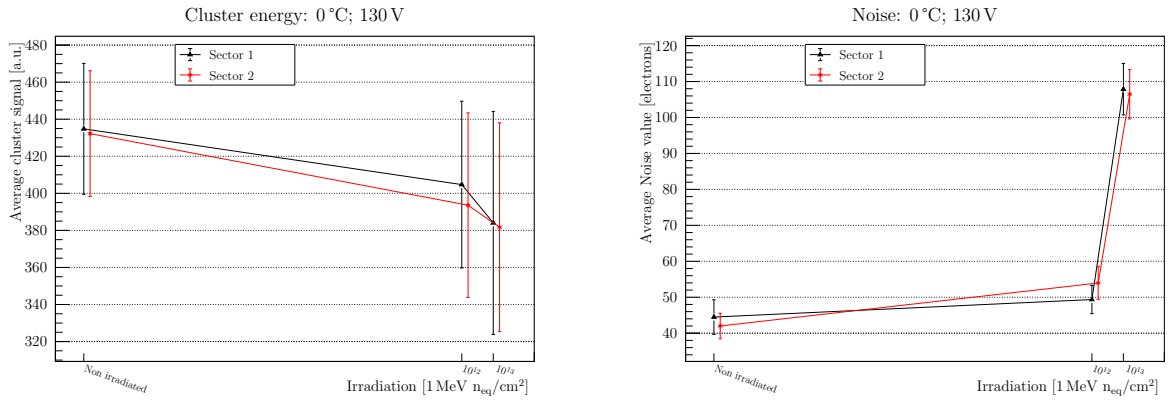
Figure 3.26: Some issues faced analysing data.

Another observed effect was the non-usability at room temperature of the devices irradiated with 1×10^{13} 1 MeV n_{eq}/cm^2. Typical failure signatures include devices where signal output was more or less constant, independent of the presence of the source in the sensor field of view; or circuits with a lower-than-expected noise (see Figure 3.26b). This effect is probably due to the fact that at room temperature and with this radiation damage, the leakage current saturates the sensor output.

Concerning the two sensors in the set irradiated with 1×10^{14} 1 MeV n_{eq}/cm^2(B4 and B5), no measurements were actually performed on them. This decision was driven by the concurrence of two reasons:

- considering that the two sensors irradiated with 1×10^{13} 1 MeV n_{eq}/cm^2) were not working properly at room temperature, and looking at how the leakage current scales with neutron fluence (see Figure A.10), the heavily damaged sensors are not expected to work properly. Indeed the ARCADIA detector is meant to work around room temperature, hence a low temperature measurement would be beyond the technical purpose of the device;
- as the whole mezzanines were irradiated, some electric components (likely the passive capacitors present) became highly radioactive. The logistic problems in storing and manipulating these mezzanines were extremely complex and annoying for safety reasons, and consequently the measurements would have required a long waiting time to be performed;

Results summary

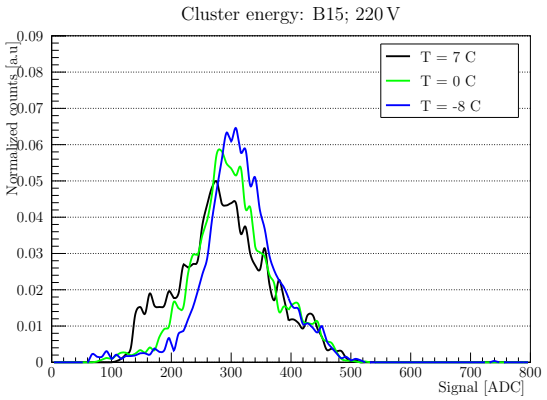


(a) Cluster energy versus neutron fluence. **(b)** Noise versus neutron fluence.

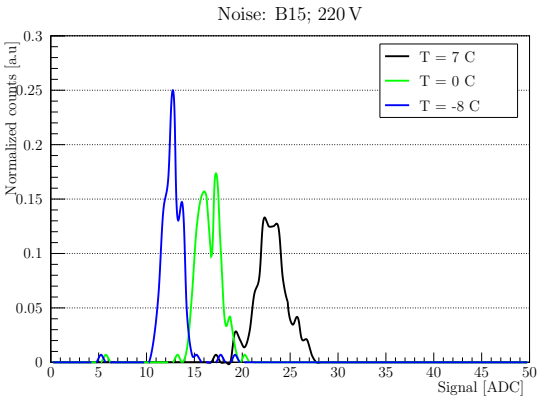
Figure 3.27: Comparison between B2, B10, and B12 sensors, first two sectors. Here, data taken at the same operating point: 0 °C and 130 V.

Here, some of the final results obtained in this experimental campaign are shown. Figure 3.27a shows the cluster energy as a function of the neutron fluence. Here the three sensors B10, B2, and B12 were used; the measurement conditions were the same in all three cases (0 °C and 130 V). Only the first two sectors were plotted as they were the most reliable ones.

Uncertainties are estimated by looking at the standard deviation of the Gaussian fit of the cluster energy. Although there is a clear decrease in the charge collected as a function of the neutron fluence, the fluctuations in the energy collected in single events are much greater. Figure 3.27b shows the noise versus the fluence; in this case the noise clearly increases a lot, especially in the most irradiated case. This increase is statistically significant. The dependence of the the performance of irradiated sensors on the temperature and the bias voltage was also studied. Only the most significant data are reported here, taken with sensor B15 (irradiated with 1×10^{13} 1 MeV n_{eq}/cm^2). Cluster energy and noise versus temperature can be seen in Figure 3.28 (constant bias 220 V); versus bias voltage can be seen in Figure 3.29 (constant temperature 0 °C). The point at room temperature (23 °C) is not reported, as the results are meaningless.

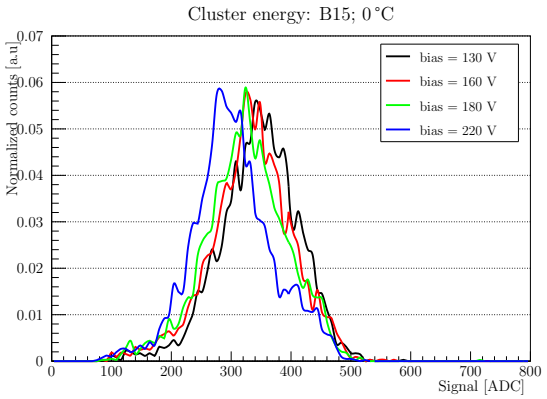


(a) Cluster energy versus temperature.

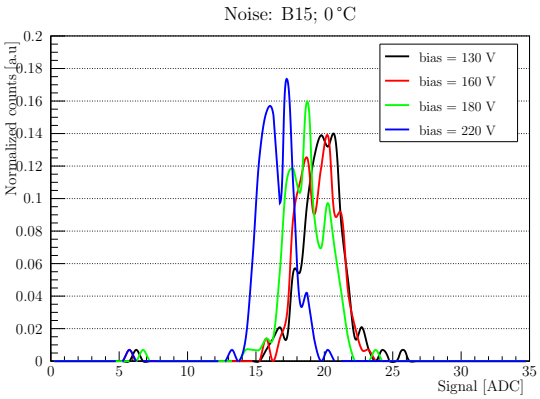


(b) Noise versus temperature.

Figure 3.28: Cluster energy and noise histograms as a function of the temperature. Here, sensor B15 (irradiated with 1×10^{13} 1 MeV n_{eq}/cm^2) and constant bias 220 V.



(a) Cluster energy versus bias voltage.



(b) Noise versus bias voltage.

Figure 3.29: Cluster energy and noise histograms as a function of the bias voltage. Here, sensor B15 (irradiated with 1×10^{13} 1 MeV n_{eq}/cm^2) and constant bias 220 V.

Looking at Figure 3.28b it is possible to confirm that the noise decreases with temperature, but three points are not enough to speculate about the trend. The little difference

in the cluster energy that may be seen by looking at Figure 3.28a, also in this case, is not significant and is probably due to the different noise values in the three cases. Regarding the bias study, for both the cluster energy of Figure 3.29a and the noise of Figure 3.29b there are no significant differences in the four cases. This means that at the lowest voltage used (130 V) the detector was fully depleted, and further increments in bias voltage do not cause improvements in the detector performance.

3.3.3 Future measurements

The results discussed in this section are in line with the expectations provided by the sensor designers, who did not expect the sensor to work efficiently at room temperature after an irradiation of 1×10^{13} 1 MeV n_{eq}/cm^2. Trend measurements (in terms of detector performance more than electrical characterisations) show that this device can be used efficiently in the intended reference environments (e.g., its performance is in line with the ALPIDE sensor requirements, which are listed in Table 2.1).

Further measurements are currently planned to evaluate the radiation resistance of this technology. Concerning the sensors that were already irradiated in the campaign described in this work, currently some tests after the defects annealing are being performed to study its effect and to compare it with the known literature (see Section A.3.3). Another test by using similar sensors was planned in 2022 with 27 MeV proton irradiations at the local SIRAD irradiation facility in Legnaro (Padova – Italy) [113] but, due to technical problems of the accelerator, it was postponed to 2023. ARCADIA sensors and test structures will be irradiated and several measurements will be performed, both similar to those discussed above, and purely electrical ones (leakage currents and CV curves). The NIEL model (described in Section A.3.1) will be used to normalise the various proton fluxes to the standard 1 MeV neutron equivalent ones. More ambitiously, if a sufficient number of ARCADIA samples are available, the collaboration might also perform neutron irradiations again at the Ljubljana reactor to remeasure the damage factor of the SIRAD protons compared to neutrons, which will provide a better benchmark measurement in the field of radiation damage.

Chapter 4

Hash readout architecture

The hash architecture is an innovative digital architecture optimised for the readout of digital MAPS with a single threshold (binary hit/no-hit fashion) achieving low power consumption and simplicity of implementation over large areas. It is designed to help in situations where, on the one hand, the particle rate is not too high (below $100\,\mathrm{MHz\,cm^{-2}}$) and, on the other hand, the constraint of power consumption is particularly strict (below $30\,\mathrm{mW\,cm^{-2}}$). It exploits an information compression architecture named OrthoPix, theorised by the Padova group [114].

The idea is to use a network (namely the **hash network**) to inform the periphery about the position of the hits within the array. The hash network implements a set of **projections** of the address of the hit pixel on opportune axes. In the periphery some machinery guesses this address by **decoding** the information contained in this network. If the fraction of hits inside the array is not too high (this value depends on the architecture implementation, as will be discussed later - as a rule of thumb, below 1 hit every 200 pixels of the array), the guesses are *mostly* right (error rate $\leq 5\%$, as is quantified in the next chapter). This approach trades the number of bits used to transmit the information and the pixel complexity with a lossy compression algorithm that may produce incorrect addresses that must be taken care of correctly; implementing simple in-pixel logic, it is possible to achieve small pixels and low power consumption.

The hash architecture can be implemented in several different flavours that use different decoders and alternative readout-and-reset routines. In the implementation detailed in this work, the pixel can be pushed to be small because of the simple readout routine, but the power consumption, even if very low, is not as good as it may be. Alternative and bolder implementations (which could lead to missing hits and ghost ones, as discussed in Section 5.4) can make this architecture fully take advantage of the hash approach, but may not be reliable for high particle rates.

This chapter presents both the methods of the analysis performed in this thesis and the theory behind the usage of the hash architecture. Section 4.1 introduces the methods used to assess the performance of a CMOS circuit and, specifically, a binary pixel readout

architecture; in Section 4.2 the hash architecture is presented in comparison with other standard and already developed architectures; in Section 4.3 the problem of hash projections is studied from a purely mathematical point of view. The actual physical implementation, together with a study on the performance of the hash architecture, are analysed in Chapter 5, where different flavours are presented.

4.1 CMOS circuits and pixel readout

The theory of CMOS circuit design is very wide, and there are several books in the literature detailing the best design techniques [46], [107], [115], [116]. In this section, only the concepts that are important for the purpose of this study are recalled. In addition, the metrics of the study that is carried out in the next section are set, allowing to compare different readout architectures implemented for tracking pixel sensors.

The focus of this thesis is MAPS; nevertheless, a similar approach could be used to describe the behaviour of a readout architecture used for hybrid devices or, even more generally, for the readout of an arbitrary array of binary values using an ASIC. A concept that is taken from granted through this chapter is the **sparsification**: information is present only in a small fraction (something of the order of one per mille) of the pixels.

4.1.1 CMOS circuits

A **C**omplementary **MOS** (CMOS) circuit is an architecture where **M**etal-**O**xide-**S**ilicon **F**ield-**E**ffect **T**ransistors (MOSFETs or MOS) are used to implement logic functions. Transistors are used as switches: An nMOS (a MOSFET where electrons are the majority carriers) lets current pass when the gate voltage is high, and a pMOS (a MOSFET where holes are the majority carriers) lets current pass when the gate voltage is low. In CMOS circuitry, the output pin is always connected only to either the reference voltage (represented as V_{DD}) or the ground. An example of the simplest CMOS circuit (an inverter) is shown in Figure 4.1.

The main advantage of CMOS circuitry, compared to the numerous alternatives, lies in the fact that the power consumption is extremely low: static power consumption is suppressed by transistor technology (where an oxide is used to separate the gate from the bulk where the channel opens), and dynamic power consumption is reduced by the fact that there are never low resistive paths between V_{DD} and ground. Another advantage is the level of integration reached by the technological development: it is possible to create extremely small transistors, with channels as long as 3 nm [70], leading to very small **I**ntegrated **C**ircuits (IC).

The approach often pursued when designing a CMOS circuit takes advantage of what are called **standard cells**: To design the circuit, not every transistor is placed singularly, varying its size and performance, but some standard cells, provided by the foundry, are used to create the circuit. Some examples of standard cells are `OR`, `AND`, `BUF`, `FLIP-FLOPS`, `MUX`,

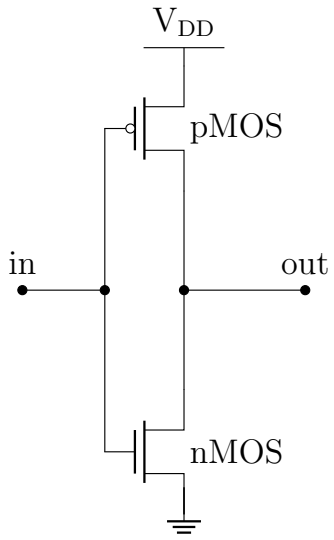


Figure 4.1: A CMOS inverter circuit. if the input is high, the nMOS conducts while the pMOS does not, hence the output is connected to ground; if the input is low the pMOS conducts and the nMOS does not, hence output is connected to V_{DD}. In no situation there is a short between V_{DD} and ground.

each capable of performing a digital operation on a signal. For every standard cell, there are different flavours, changing the size of the transistors used. For example, by changing the type of `OR` cell used, it is possible to trade a small cell with low power consumption for a larger cell with better timing performance. This approach is easier (as there is no need for intensive SPICE[1] simulations to determine the circuit behaviour), but usually less performing, as standard cells are multipurpose and are not optimised for the given task they need to achieve in the particular circuit. All standard cells can be found in a database that is called **technological library** (or just library), and it is provided by the foundry, given the technology node. The library comes with several tables that summarise the performance of each standard cell. In this work, the following quantities from the library are used:

- input capacitance of the cell;
- maximum output capacitance that can be driven by such cell;
- leakage power for the cell;
- propagation delay, both for rise and fall of the input, as a function of the load capacitance and the input transition;
- output transition, both for rise and fall of the input, as a function of the load capacitance and the input transition;
- dynamic power consumption, both for rise and fall of the input, as a function of the load capacitance and the input transition;

Additionally, for every standard cell, there are different tables for different temperature and reference voltage corners. On top of the standard cells, the technological library also provides interconnections among them: there is a fixed number of metal layers, each one of them has

[1] **S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis; a software used to extrapolate circuital performances.

a proper resistance and geometrical constraints (like distance between neighbouring lines), and they are connected via proper vias. Concerning the hash architecture, the reference technological node is a modified 110 nm commercial CMOS process from LFoundry [30], already introduced in the discussion of ARCADIA sensors (Chapter 3). The technological library is covered by **non-disclosure agreements** between the research group developing the CMOS circuit and the foundry providing the library; therefore, throughout this work, the details of the methods used to calculate the various delays, gate size, and power consumption are not made explicit; only the final results are presented.

These quantities are used to predict both the power consumption and the time delay introduced by a CMOS circuit; thus making a comparison between different readout architectures possible. In the following pages, a description of how to quantify these circuital properties is reported.

Power consumption

In a circuit as the one in Figure 4.1, as well as in more intricate ones, energy is consumed as a result of various factors and power consumption is generally divided into **static** or **dynamic**. Static consumption (also known as steady consumption) is the energy consumed just because the device is powered; dynamic consumption, on the other hand, is caused by the voltage changes inside the circuit and, therefore, is related to the operation of the circuit itself, seen as modification of an input signal. Although static power consumption is considered mostly negligible in comparison to dynamic one, the continuous shrink of the transistor size, pushed by technological advancement, makes this contribution non-negligible. This is particularly true in circuits like the ones discussed in this work where the clock is not propagated to the biggest part of the IC, hence the dynamic power consumption drops. In Figure 4.2 it is possible to see the trends in static and dynamic power consumption in the modern CMOS technology node as a function of year and transistor size, considering an example processing unit. At 110 nm the dynamic power dissipation is still dominant over its static counterpart in these commercial devices.

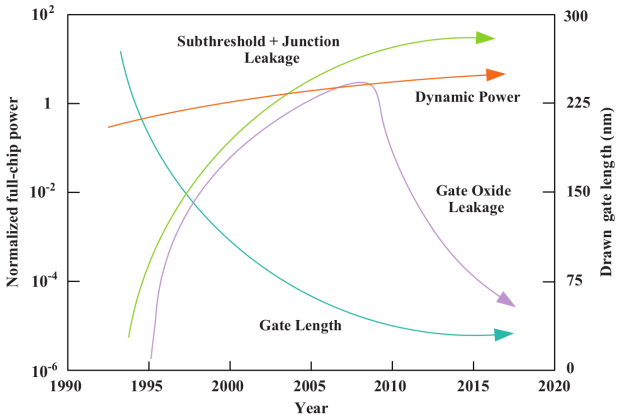


Figure 4.2: Total chip static and dynamic dissipation trends. Taken from [117].

The most important sources of power consumption are the following ones [118]:

- power due to **diode reverse biasing**: there are always reversed biased diodes in a transistor, either between the bulk and the drain or between the bulk and the source. These diodes have a reverse current flowing, although very low. This is a static and negligible source of power consumption;
- power due to **gate current**: Despite the fact that the gate is separated from the bulk by an oxide, this oxide is very small in sub-micron technologies (nanometer scale [119]), and may contain some defects. Therefore, there is a small current that flows through the oxide. This source is static as well and very often non-negligible for low clock periods;
- power due to **subthreshold current**: this effect is due to current flowing through the switched off transistor, as there is a path between V_{DD} and ground through a transistor that is nominally switched off. Also this effect is static and is considered negligible in first approximation when compared to the other sources;
- power due to **loading of capacitances**: downstream a circuit like the one in Figure 4.1 there is a load capacitance that gets loaded and unloaded depending on the circuit output state. During loading operations, the initial voltage was 0 and the final voltage is V_{DD}. Half of the energy spent is stored in the capacitance (E_C) and half is dissipated in the pMOS circuit[2] (E_r). Both the energy dissipated and the energy stored in the capacitance have a value of:

  $$E_r = E_C = \frac{1}{2} C_L V_{DD}^2$$

  However, during the discharge of the circuit, the energy that was stored in the capacitance is dissipated along the nMOS circuitry. Therefore, for a full transition, the energy consumption (E_T) is

  $$E_T = C_L V_{DD}^2 \tag{4.1}$$

  and is consumed half in the pull-up network (on the pMOS side) and half in the pull-down network (on the nMOS side). This type of power consumption is dynamic, and it grows linearly with the number of transitions per time unit, called **switching activity**. This component is dominant for circuits working at relatively high speed;
- power due to **direct-path currents**: both nMOS and pMOS have finite switching time between their on- and off-states; hence, there can be moments in the circuit where there is a direct path between a conducting nMOS and a conducting pMOS. This effect is also dynamic, as it depends on the number of transitions but is smaller than the previous one (approximately 20% [107]).

There is also heat dissipated through wires with finite resistance, but in a CMOS circuit the connection is always between transistors, hence everything can be interpreted as capacitance charging and discharging. This means that the resistance of the wire does not affect the power consumption in first approximation.

[2]The energy dissipation in a circuit charging a capacitor is completely independent from the resistance.

Time delays

In a CMOS circuit such as the inverter in Figure 4.1, delay is defined as the time it takes for a signal to reach the output after a modification in the input. To estimate it, it is necessary to take into account:

- The time taken for a voltage change to travel across a wire, depending on the resistance and capacitance of the wire. The following formula is used:

$$t_d = 0.7RC \tag{4.2}$$

  taken from [120]. The resistance and capacitance of a wire can be estimated by studying the reference technological library and an average crowding for the metal layers used;
- The delay introduced by the presence of a standard cell. This delay depends on the cell used, on the input transition (i.e., rise and fall time), and on the capacitance to drive, and can be found by analysing the design and looking at the technological library.

It is worth noting that, while the power consumption is independent of the resistance of the interconnections, the delay actually is strongly dependent on this physical quantity.

The formula for the delay actually hides the need to segment the long connections between the cells. Consider a long interconnection. In first approximation, both its resistance and capacitance are proportional to its length; hence, the delay has a quadratic dependence. This means that by segmenting such a connection and interspersing the segments with buffers, it is possible to reduce the total delay of the line, even if every buffer adds some delay. In contrast, the power does not depend on the resistance: this segmentation leads to an improvement from a timing point of view paid for by the buffer power consumption and the occupied area. In this work, the buffering scheme will be fixed (i.e., buffers are placed in long lanes keeping constant the distance between two subsequent buffers), making the time actually linearly proportional to the interconnection length. Some more detailed studies are reported in Section 5.2.

4.1.2 A readout circuit

The discussion in the last paragraph is applicable to any generic CMOS circuit. In this work, the examined circuits are pixel readout CMOS circuits. Formally, they are defined here as "CMOS circuits capable of physically moving an information from an arbitrary position in an array (called pixel) to a fixed position (called periphery), reconstructing the original address of this information as well". Usually, the functioning of these circuits can be divided in two subsequent phases:

- a **readout phase** where the information is copied from the pixel to the periphery and the address of the original position is registered;
- a **reset phase** where the information is erased from the pixel.

In this work sparsely populated arrays (as previously defined) are considered: information can be found at the same time on a limited fraction of pixels, below 1 per mille.

In order to compare effectively different readout algorithms it is necessary to find a way to translate the CMOS circuital properties listed before to a quantification of delay and power consumption in a readout architecture. The figures of merit that are compared quantitatively in this work are the energy consumption per readout, the time per readout, and the problem of many-hit clogging, peculiar of some readout systems such as the hash one. Moreover, estimates of the static power consumption and total area of the hash architecture are carried out as well.

A first property characterising pixel-readout systems is their anisotropy: there is a preferred direction in the array. In most cases, the information moves along one direction only (the **column**), and each column can operate in parallel (more details on this are given in Section 4.2). This approach is called **column readout**. As a consequence, there is a preferred direction for wiring, conventionally called the **vertical** direction.

Time per readout

The time required to evacuate a single hit is one of the most important parameters when assessing the performance of a readout architecture, as it directly impacts:

- The **timing resolution**: the more it takes to evacuate a hit, the less precise the timestamp associated to it will be. Timestamp precision is particularly important for some applications (such as proton tomography, see Section 2.4.4);
- the **efficiency**: the slower the hits are evacuated from the array, the higher the pile-up probability, as a second particle might hit a pixel that already contains data. This undermines the sensor efficiency;
- the **maximum rate capability**: When data arrive continuously, the faster the sensor can process hits, the higher the particle flux can be. This can also be seen as an *efficiency* effect as, with increasing particle rate, efficiency eventually approaches zero as all pixels are already populated.

This quantity has two major contributors: the clock frequency and the readout mechanism itself. **Clock** is a particular signal used in every CMOS circuit used for the synchronisation of signals. In some circuits (like the architectures analysed here), it is used only for part of the logic. The clock is a signal that oscillates, characterised by its period (the time between two subsequent positive edges) and its duty cycle (the fraction of the period the clock spends in the *high* state, usually 50%). Considering that some synchronisation techniques are necessary to properly read and transmit the data, clock is always necessary. The lower the clock period, the lower the readout time (and, considering that the overall power consumption strongly depends on the switching activity, the higher the dynamic power consumption). The second contribution is the readout mechanism *per se*: some digital logic allows to read several hits in the same clock cycle, some others need several clock cycles to read one hit.

Energy per readout

As each action performed by a digital circuit comes with an energy cost, it is important to find the most important contributions.

Along the vertical direction, the interconnections are longer; therefore, whenever a bit changes its value in one of these interconnections, a lot of energy is consumed[3]. These bit changes are called **long-lane transitions** throught this work. By comparing the number of long-lane transitions per architecture, it is possible to compare the different dynamic energy consumption per readout in the different architectures. This approach was already followed by ALPIDE collaboration in [120].

In addition to this contribution to the power consumption, it is necessary to take into account the static power consumption of the transistors. It can be considered as depending on the transistors number and size, and can be estimated via proper Monte Carlo simulations. It requires a complete simulation after the place-and-route to be deemed reliable, especially in the most crowded regions of the circuit, such as the pixels themselves or the readout unit, where this value is higher. In this work, the power consumption of the pixel was calculated in *post-synthesis* (and therefore after the choice of the optimal standard cells to use) and before the place-and-route, and no estimations were performed on the power consumption in the periphery. It is possible to consider that the power consumption in the periphery does not depend strongly on the array readout paradigm; e.g., the power consumption of a serializer for the data transmission is not dependent on how those data reached the edge of the sensor.

Many-hit clogging

In some architectures (this is the case of the hash architecture, analysed below) both the timing and the power performances depend on the current particle flux, as some processes are triggered only when the flux is particularly high and the number of hits in the array is above a given threshold. Although this is not a problem for many pixel readout architectures, it is considered in this work to make the comparison with the hash architecture feasible.

Quantitatively, the benchmark set for this quantity is a performances loss of 5%. For any architecture discussed below, the maximum flux is defined as the highest one such that the energy-per-readout and the time-per-readout (on average) stays within the 5% of the optimal performances that are obtained when the hit rate approaches zero.

[3]This quantity is not reduced, but, on the contrary, increased, by the long-lanes segmentation necessary to achieve good timing performance.

4.2 Three readout architectures

In this section, three different readout architectures are compared. The first one, clockless column readout, is a simple, abstract readout architecture idea with many limitations (Section 4.2.1). A concrete implementation that solves the paramount issues of the abstract version is the readout of the ALPIDE sensor [120] (Section 4.2.2); a reference MAPS. After these two, the idea behind the hash architecture is introduced, revealing its driving principles (Section 4.2.3). The pros and cons of each architecture introduced are listed and some semi-quantitative comparisons, performed according to the methods unfolded in Section 4.1, are performed.

As mentioned previously, the device taken as a reference is a pixel sensor in binary hit/no-hit fashion. This means that the only information stored in the array, and to retrieve, is whether or not a pixel was hit, and there is no information concerning the amount of energy collected. Also the sparsification hypothesis is taken as granted.

For the sake of simplicity, throughout this section a reference shape for the array is set. The sensor considered is a square $N \times N$ array: every pixel has an address that can be written, using standard bits representation, using $\texttt{ceil}[\log_2(N \times N)]$ bits[4]. For physical convenience, N is thought as a power of 2, optimising the usage of the available lines.

4.2.1 A clockless column readout

The first architecture discussed, taken as a simple reference useful to understand the analysis performed, is called **clockless column readout**. It is an architecture where the information moves vertically along columns, and the clock is not propagated through the array in order to save power. At the end of the column, a synchronous **Readout Unit** (often referred to as **Periphery**) manages data ordering and serialisation toward the **Output** (see Figure 4.4). It is an example of a column readout architecture, with a preferred direction.

In order to understand how this architecture works, it is necessary to start by describing a readout routine. The simplest routine is performed whenever there is a hit in a pixel and there are no other hits in the same column. In this case, the hit pixel writes its row address on a shared bus through the column to which the pixel belongs. This row address travels through the array until it reaches the periphery, where this information is read (readout phase). The bi-dimensional position of the hit pixel is reconstructed by looking both at the read information and at the position of the bus that brought the information: the former is used for the y (the vertical position, the row address), while the latter is used for the x (the horizontal position, the column address). Whenever the periphery reads a hit, a reset signal is broadcast back to erase the information inside the array (reset phase).

For this readout routine to work properly even when there are several hits distributed in the array, it is necessary for the pixel to know whether it can use the shared bus or not; i.e., implement some sort of **bus arbitration**. The easiest implementation consists of a simple

[4] `ceil` is the ceiling operation, that returns the closest non-lower integer given a real number.

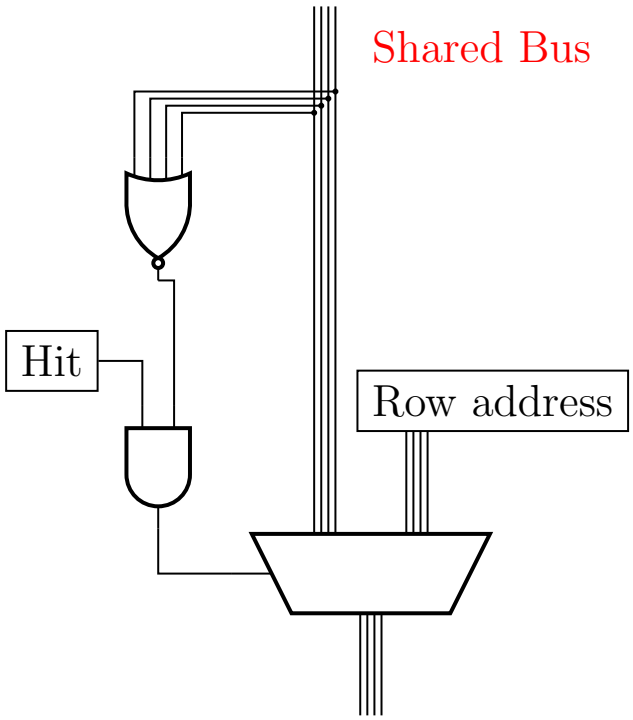


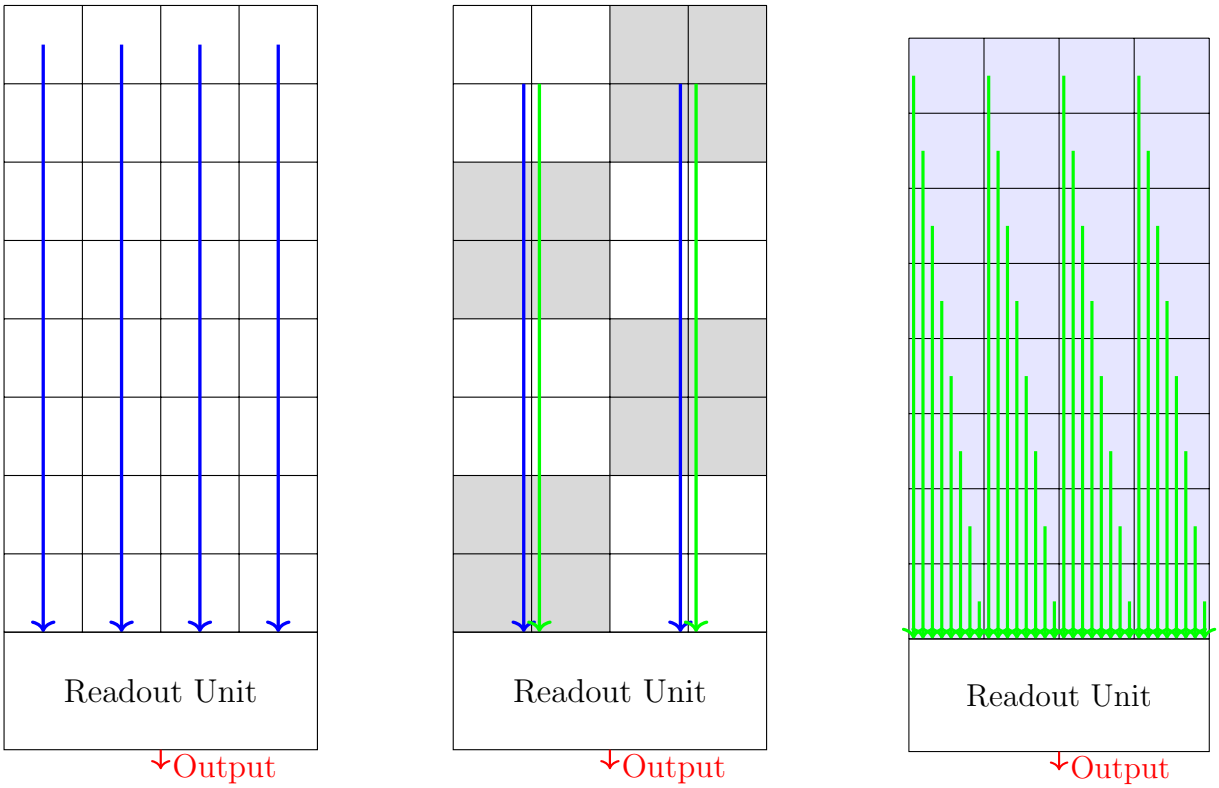
Figure 4.3: Simple bus arbitration for the standard flavour of clockless column readout. The internal row address, saved in-pixel, is broadcast only if `Hit` value is one and there are no high bits in the row address bus. Here, implementation with a multiplexer, an `OR` gate and an `AND` gate. This circuit is repeated for every pixel, and the vertical lines represent the bus shared among pixels in the same column.

check performed before the data writing: if at the pixel height there are data in the shared bus, then the pixel waits for it to be empty, otherwise it immediately starts communicating. This means that the highest pixels have priority over the pixel closest to the periphery. A scheme of such logic can be seen in Figure 4.3.

If N is the height in pixels of the array itself, the width of the address bus running vertically must be $\texttt{ceil}(\log_2(N))$ in order to accommodate all the row addresses. Then, the reset signal can travel on one single line traversing all the column.

The clockless column readout comes in three flavours: standard, clustering, and framing. What was discussed in the previous paragraph is the standard implementation, where only the row address is communicated to the periphery, and only the pixels actually containing the data are read. This approach takes maximum advantage of the sparsification hypothesis and performs the most advanced zero-suppression approach possible, as pixels that are not hit do not communicate any information. Clustering algorithm, on the other hand, tries to reduce the number of readouts per physical event, diminishing the zero-suppression. This can be useful as an impinging particle often creates a signal on several adjacent pixels. In a clustering architecture, pixels are grouped in **super-pixels** (or **pixel regions**) and the entire super-pixel is read out at the same time. In this case, the replicated unit is the super-pixel, and it is at this level that bus arbitration is performed: the number of transistors is

hence reduced as well. An example of clustering can be seen in Figure 4.4b, where the same colour represents a single super-pixel made of 2×2 pixels.



(a) Standard readout. **(b)** Clustering readout. **(c)** Framing readout.

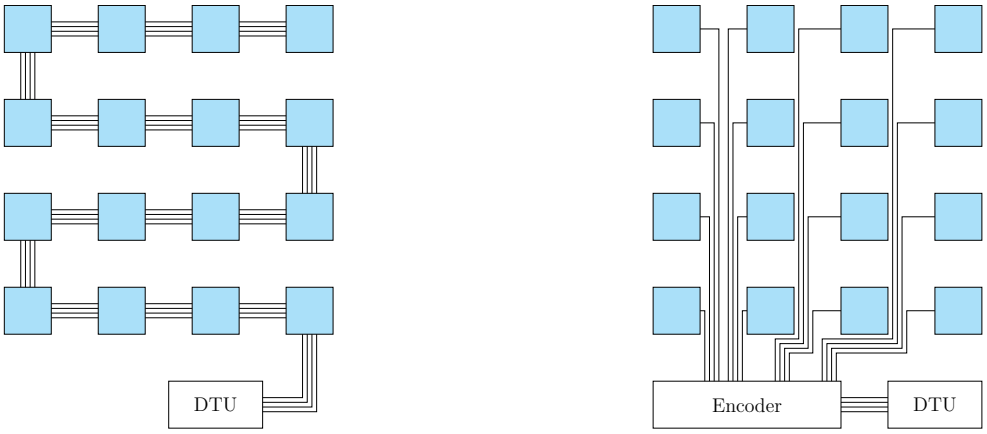
Figure 4.4: In column readout information travels only vertically. At the end, a Readout Unit serializes data toward the output. Blue lines are for the row address, green ones for data (the hitmap). More details in the text.

In a clustering architecture there are fewer addressing lines, as there are fewer super-pixels than pixels, but the row address is no longer sufficient information: the **hitmap** of the super-pixel must also be communicated to the periphery. It is a word containing, in one-hot encoding, the map of the pixels hits values. In this case, the number of vertical lines necessary includes the addressing lanes (scaling as $\log_2 N$) and hitmap ones (a fixed value of M, the number of pixels in a super-pixel). A super-pixel is read out when at least one hit is present in one of its pixels. The readout mechanism is exactly the same as described before in the **standard** flavour of this architecture. The same reset signal is propagated to all pixels in the super-pixel.

The clustering concept can be extended to the limit: the whole array is seen as a single super-pixel. This is the case of **framing** detectors: when an external trigger arrives, or when there is at least one hit in the array, the whole array is readout at the same time. In Figure 4.4c this architecture is represented. In this case there is no need for addressing, but the complete hitmap is flushed toward the array periphery, and by knowing the physical position of the lane it is possible to reconstruct the hit position, both x and y. This flavour

does not take advantage of the sparsification hypothesis, and does not implement any sort of zero-suppression. Concerning the pixels reset in this architecture, the same reset signal is propagated to all pixels in the array, erasing the information in all the pixels.

The difference between these three flavours lies in when the **encoding** operation is performed. Encoding refers to the necessary operation that produces the actual address of the hit pixel among all the possible ones. Focusing on one column only, the standard flavour implements what is called **early encoding**, as the address is written in the bus in binary representation; while the framing flavour implements **late encoding**; i.e., the address must be reconstructed looking at the position of the lane that brings information. A representation of the difference between early encoding and late encoding can be seen in Figure 4.5.



(a) In early encoding, the bus is shared, and the address is trasimtted to the **D**evice **T**ransmission **U**nit (DTU).

(b) In late encoding, all pixels are connected to an encoder, that produces the address.

Figure 4.5: A scheme representing the difference between early and late encoding.

Power considerations

According to the methods described in Section 4.1.2, to assess the power consumed by such an architecture, it is necessary to count the number of long-lane transitions. In this architecture these transitions are forced on vertical lanes only, as there are no horizontal lanes used for the array readout. The transitions count begins at the moment when the array is empty before a new hit and ends at the moment when the array is once again empty after resetting the hit pixel. This count is repeated for the three flavours introduced, and the comparison is semiquantitative; the results in Table 4.1 are computed using elementary algebra. Here, the clustering readout refers to an architecture implementing super-pixels made by 2×2 pixels, for the row address communication the average number is shown, and only one transition is considered per long lane (i.e., only the passage from low to high).

The numbers shown here cannot be used for a complete power estimation (details in Section 4.1), but provide information only on the power consumed for communication, not for sensor functioning *per se*. More details are given after the presentation of the physical implementation and of the Monte Carlo simulations of the hash architecture in Chapter 6.

1 hit in the array

	standard	clustering	framing
Address communication	$\frac{1}{2}\log_2(N)$	$\frac{1}{4}\log_2(N)$	-
Hitmap communication	-	1	1
Reset request	1	1	1
Total	$1+\frac{1}{2}\log_2(N)$	$2+\frac{1}{4}\log_2(N)$	2

2 hits in the array (same cluster)

	standard	clustering	framing
Address communication	$\log_2(N)$	$\frac{1}{4}\log_2(N)$	-
Hitmap communication	-	2	2
Reset request	2	1	1
Total	$2+\log_2(N))$	$3+\frac{1}{4}\log_2(N)$	3

H hits in C clusters

	standard	clustering	framing
Address communication	$\frac{H}{2}\log_2(N)$	$\frac{C}{4}\log_2(N)$	-
Hitmap communication	-	H	H
Reset request	H	C	1
Total	$H(1+\frac{1}{2}\log_2(N))$	$H+C+\frac{C}{4}\log_2(N)$	H + 1

Table 4.1: Vertical long-lane transitions in the three flavours of clockless column readout architecture described in text.

Time considerations

Concerning the time that is necessary to perform a readout, a precise estimation cannot be computed on such an abstract readout architecture idea. Nevertheless, a comparison among the different flavour is possible. For this purpose, k is defined as the average necessary time for one readout routine as previously described, and it includes:

- The travelling time the address information takes to reach the periphery, directly proportional to the height of the pixel inside the array (these are the longest wires with the longest delay);
- the time the periphery takes to read the address and request a reset, directly proportional to the period of the clock running in the periphery;
- the travelling time once again as the reset signal must reach the pixel containing the signal to ask for its reset.

Here, k is considered constant with a simplistic approach, as there is no guarantee that the speed of the signals is constant (because of different parasitic capacitance), and the approach that periphery uses to sample is not modeled, as it would need further details.

Then, the evacuation time trend as a function of the population value of the array can be calculated depending on the parameter k. The results, calculated once again using elementary algebra, are reported in Table 4.2.

Number of hits	standard	clustering	framing
1 hit	k	k	k
2 hits (same cluster)	$2k$	k	k
H hits in C clusters	Hk	Ck	k

Table 4.2: Time needed to read a given number of hits in a clockless readout array with different flavours. k is the time needed for a complete readout cycle.

Array many-hit clogging

Many-hit clogging is defined as a worsening of the sensor performance due to the very high fraction of pixels currently containing a hit. To analyse the possible issues arising, it is necessary to study the case when a second hit reaches a column where a first hit was already present. This analysis is qualitative, as several more details would be necessary to completely interpret what happens inside the architecture, which is left abstract for the purposes of this work.

- If a hit appears below a pixel currently communicating its address, the bus is not seen as empty; hence, the second pixel hit knows that the column is currently busy in a readout. Thanks to the bus arbitration scheme (Figure 4.3) this second pixel does not superimpose on the readout that is currently ongoing. When the first readout is over, and the higher pixel is reset, the second pixel starts communicating its address in the shared bus. In this case, there are no issues from nor from the energy per readout nor from the time per readout point of view. Moreover, the two hits are communicated correctly to the periphery;
- If a hit is generated above a pixel currently communicating its address, the second hit pixel is not aware of the fact that there is an ongoing readout. For the following argument, pixel A refers to the first and lowest pixel hit, whereas pixel B is the second and highest one. This means that B starts to broadcast its address on a line currently driven by A. What happens in this case is actually strongly dependent on the logic implementation of the architecture. One of the safest option is to ask A to stop broadcasting its address as soon as the new signal arrives on the shared bus. This means that the periphery can sample A or B and, similarly, the reset can be executed by either A or B. After this, in the best case, the same hit (either A or B) is read and reset; in the worst case, on the other hand, one of the hits is read and the other

is erased. This may lead to a missing hit (a hit that was incorrectly erased) and/or a duplicate hit (a hit that is erroneously read twice). Nevertheless, during this process, there is not a major waste of power, nor of time; hence, many-hit clogging as it was defined is not a paramount problem of this abstract architecture.

Flavour choice

This kind of architecture, with its three flavours, can be taken as a reference, and the flavour choice depends on the physical application. It is worth noting that out of all three, the framing flavour is quite expensive in terms of physical space, as there is the need to connect every single pixel to the array periphery, a price that may be alleviated by using relatively large pixels (to host all the transistors, in a technology node like the one taken as a reference at 110 nm, and a reference size around 1 cm, a pixel bigger than 100 µm) or using alternative architectures (like the CMOS sensors currently used for imaging).

Architecture limits

One of the main issues that is encountered when this architecture is implemented is how to access the address of the row inside the pixel. In Figure 4.3 there is a box, labelled *row address*, that represents the height the pixel. This means that in the physical implementation there must be some customised vias in every pixel connecting the bits either to zero or one.

More importantly, there is a major issue related to collisions: in the development of this architecture, some time may pass between the start of the address communication and the hit erasing in the proper pixel. This may lead to inefficiencies like the one highlighted in the many-hit clogging analysis for this architecture. Although this inefficiency does not create issues in terms of neither energy nor time for a readout, it makes the device highly unreliable. It is for this reason that this very simple architecture, described as an introduction to readout architectures, does not match an actual implementation in a MAPS, that host safer and more performing readout architectures, like the ALPIDE one described in the next section.

4.2.2 ALPIDE digital architecture

A concrete implementation of a clockless column readout architecture, addressing the limiting problems of the abstract architecture discussed above, can be found in the ALPIDE detector (already introduced in Section 2.4.1). The readout architecture of the ALPIDE sensor is collision safe and maintains optimal efficiencies and a very low fake hit rate up to low-mid impinging particle rates (see Table 2.1). The description of the readout architecture reported here is taken from [120]. The goal of this architecture is to achieve low power consumption (below 100 mW cm^{-2}) and fast readout time (the time required to read one single pixel is approximately 25 ns) [89].

ALPIDE chip implements a tree-like structure for the row address communication: the

building block is the AERD circuit (**A**ddress **E**ncoding **R**eset **D**ecoding) which is repeated in a hierarchical fashion to transmit the address towards the periphery (and the reset towards the pixel the other way around). The building block scheme is reported in Figure 4.6 and is used to reconstruct the row position of the hit pixel inside the matrix. This approach is particularly low-power-consuming. In Figure 4.7 such a tree is shown. The number of layers in the tree and the bus width are selected to optimise the delay, the power consumption, and the overall number of transistors.

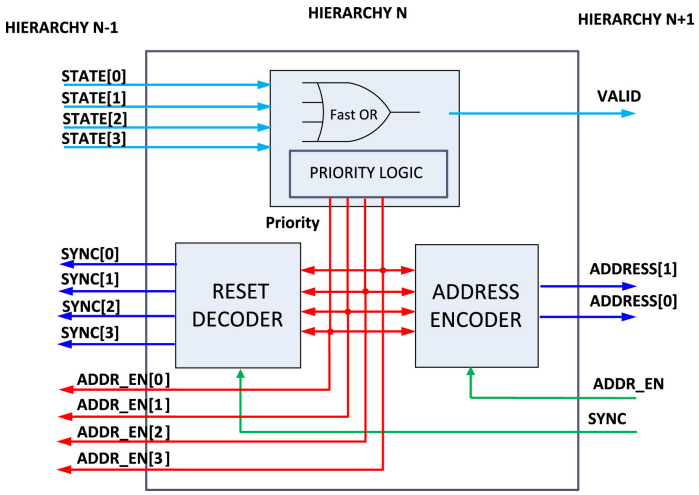


Figure 4.6: AERD system as implemented in ALPIDE chip. Taken from [120].

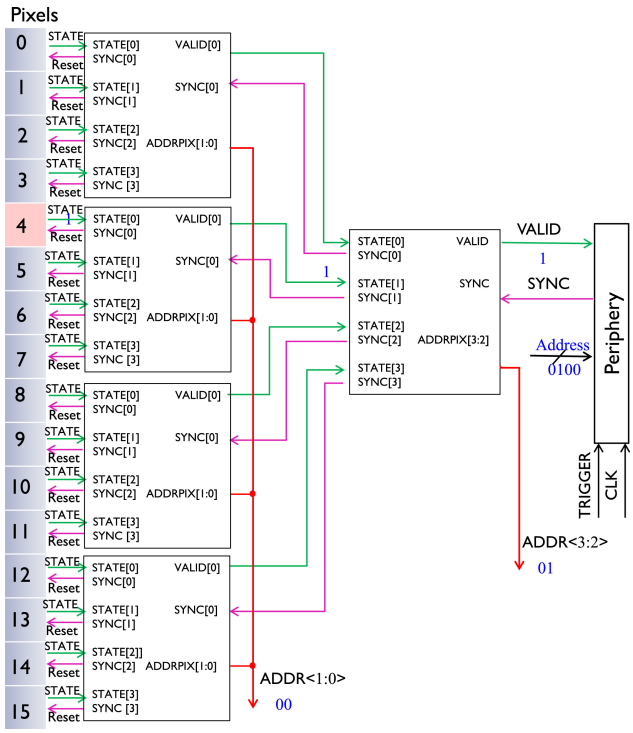


Figure 4.7: A tree of two AERD layers to reconstruct the address of 16 pixels (address can be read, in blue, at the bottom). Taken from [120].

The priority logic in the array works as follows: output labelled ADDRESS is set to be the address of the **lowest** high bit in the input bus STATE. the hits closer to the periphery are always emptied before the farther ones. This ensures that there are no collisions between hits, a major problem in the architecture described before, as only the address of one pixel is successfully produced by the chain.

Technically, the ALPIDE chip is not, by default, in data push mode; i.e., the information from the analogue front end is registered only if there is an external signal, called STROBE, that is high. Pixels are logically and physically grouped in double columns formed by two adjacent 512-pixel high columns sharing the same readout tree and periphery logic, and each double column can independently perform a readout at the same time (an example of column readout). The readout of a single pixel works as follows: first, a VALID signal is generated in the pixel using a fast OR gate. Its arrival at the periphery triggers the generation of a signal called SYNC, broadcast through the column. Only the pixel selected by the priority logic responds to SYNC starting to transmit its address using the address encoder. Then, on the falling edge of the SYNC signal, the pixel is reset. The periphery samples the address at the falling edge of the clock signal (in the periphery). The readout timing diagram can be seen in Figure 4.8.

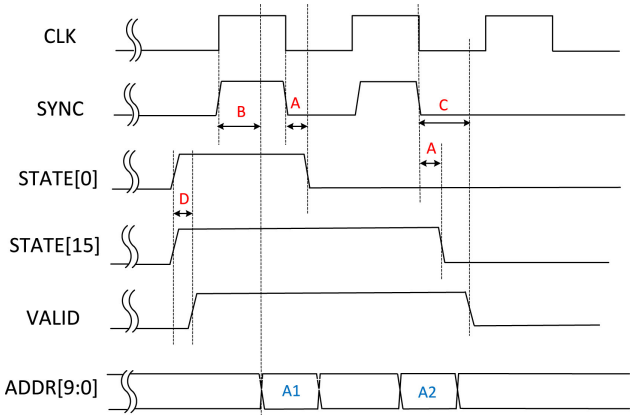


Figure 4.8: Timing diagram for the readout of ALPIDE array. Here two pixels contain information: 0 and 15. Taken from [120].

In the ALPIDE sensor the encoding is **hierarchical**, which is a mix between early and late encoding. For every step in the replicated hierarchical tree, some bits are added to the hit address. This means that the encoder is actually *distributed* through the array.

Physical area considerations

An estimation on the area of the AERD circuit can be found in [120], where an image of the design is shown (and reported in Figure 4.9).

From this figure, it is possible to estimate the overall area used for the digital architecture of four pixels (the ones in figure) to be of the order of 1000 μm^2.

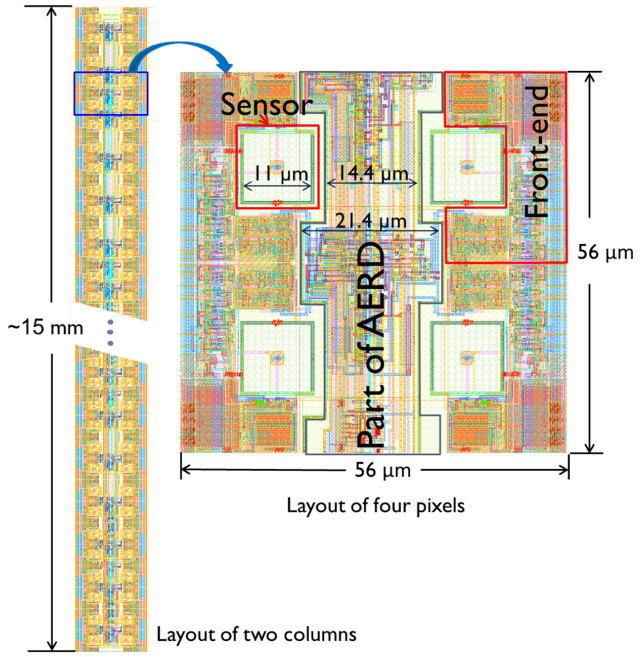


Figure 4.9: An image of the ALPIDE layout. Taken from [120].

Power considerations

Some simulations have been performed after place-and-route on the ALPIDE sensor to estimate the total power consumption, and their result can be seen in Figure 4.10. Here, the green part is the component due to the long lane transitions discussed here, and the red part is due to the various idle power consumption sources. These two contributions are the ones that are due to the readout architecture chosen, and can be reduced by changing the paradigm, as discussed in this work.

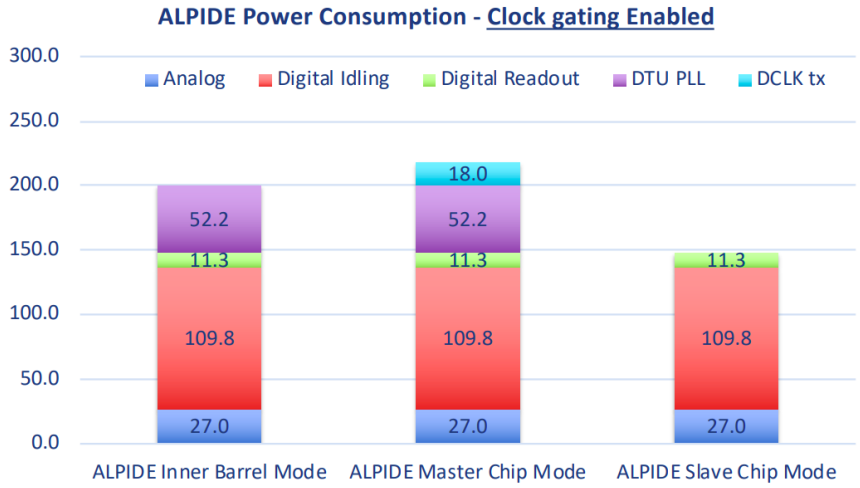


Figure 4.10: Simulations on typical power consumption in ALPIDE. Taken from [29].

The dynamic contribution of the digital readout can be estimated by counting the tran-

sistions. In this architecture, to read a pixel, there is a long lane transition (VALID) signal to acknowledge the new data, another one in the other direction (SYNC) to request readout and reset, and then the address starts moving from the pixel toward the periphery. This means that for a readout there is the need, on average, for:

$$2 + \frac{1}{2}\log_2(N)$$

Long-lane transitions, one more than the naive architecture discussed and reported in Table 4.1, and this greatly improves the timing performances of the sensor.

The estimate of dynamic power consumption in ALPIDE sensor just discussed was carried out in [120] with the exact same method that was borrowed for the other readout architectures discussed in this work. The published average energy consumption per hit is 72 pJ. Experimental tests [54] show that the energy needed to encode the address of a hit pixel is approximately 100 pJ; the fact that these values are close gives confirmation that the method used can effectively estimate the dynamic power consumption of a MAPS.

Timing considerations

For what concerns timing, the diagram in Figure 4.8 shows how every clock cycle can be used to read one data from a column. However, the trigger at both the rising and falling edges of the SYNC signal constrains the clock period, as explained in [120]: by taking into account the delays due to the propagation of signals in the hierarchical structure (in the worst timing corner), the maximum clock period is 10 MHz. This is the theoretical highest hit rate that can be readout from a double-column; the value actually measured is mostly limited by the max capacity of the chip output link, and is of the order of 6 MHz cm^{-2} [29].

Array many-hit clogging

The best advantage of ALPIDE is its resistance to collisions. The fact that the VALID signal is broadcast before the address signal, together with the fact that the address bus arbitration is performed passively in a tree that always shows the periphery at most one address, solves the potential inefficiencies discussed in the clockless column readout. The readout logic is therefore very stable respect to the hit rate variations, and this architecture does not show any issue for many-hit clogging.

Limitations

This sensor is currently used in several experiments: from high-energy physics [54], to space physics [121], and medical applications [122]; thus, it is considered the gold standard in the field of MAPS.

Nevertheless, there are some peculiarities that limit its possible further usages from the point of view of the readout architecture:

- The pixel is relatively complicated: There is a need for an address encoder and a reset decoder for every pixel in the array. This limits the downsizing of the pixel, that must be large enough to host the necessary transistors;
- The readout tree layout is finely tuned, making the architecture hardly scalable for bigger sensors, and for changes of the technology node.

4.2.3 Hash architecture

The hash architecture is a tentative to simplify the in-pixel logic and reduce the power consumption in a MAPS like the ALPIDE, while maintaining good hits-collisions management. The idea is to skip both the address encoding and the reset decoding by implementing a system where the pixels can inform the periphery about the presence of a new hit by changing the state of a set of metal wires. The periphery uses a **guessing** algorithm to retrieve the address of the pixel and queries it directly for readout and reset. If the guesses are *mostly* right, the architecture can reach performance close to ALPIDE one with simpler pixel that may consume less power (more details in Section 5.3 in next chapter). A timeline representing a hash readout is shown in Figure 4.11.

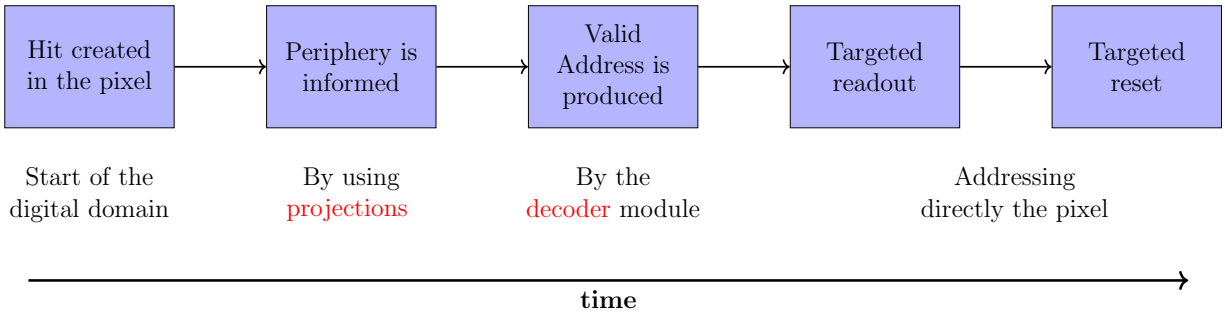


Figure 4.11: A representation of the timeline of a readout in the hash architecture.

A proper mechanism must be implemented for the periphery to guess the correct address. This is achieved through the use of **projections** of the addresses on geometrical axes. As an example, if both the x and the y projection of a hit pixel was available, it would always be possible to reconstruct correctly the address of the single hit but, in the case of more than one hit, the multi-hit degeneracy problem produces addresses likely to be wrong (data collision). This issue can be mitigated by using additional projections, that reduce the number of addresses generated by the decoding algorithm. The projection approach has already been widely used in particle detectors, such as wire chambers and strip detectors [123]. For the sake of simplicity, the sensor considered has a square shape of a $N \times N$ array. The upcoming chapter delves into the application of this particular concept to a rectangular arrangement.

To describe the idea behind the hash architecture, it is convenient to start by formally defining some concepts. A **hash word** is a set of bits obtained by projecting a position inside the array (a pixel address) on one specific axis. For symmetry reasons, each hash word is considered with the same number of bits and, by analogy with the usual projections x and

y, this number of bits is considered equal to the side of the array N. The architecture then implements a given number of projections (with only x and y there would be 2 projections), and the **hash code** is defined as the set of hash words, one for every projection. The hash code is actually a function of the current status of the array; i.e. of where the hits are. These projections are physically implemented as chains of OR gates that pass through pixels, as illustrated in Figure 4.12. In this way, in the periphery, the hash bit is 1 if at least one of the pixels in the chain actually contains data (a more realistic physical view of this is discussed in Section 5.1). The pixel is far simpler than the one discussed in the previous section: in this hash architecture the pixel does not know its own address but only sets the hash code and responds to external signals, without the need for encoding and decoding. The readout mechanism is discussed in more detail in the next chapter, where the actual physical implementation is presented.

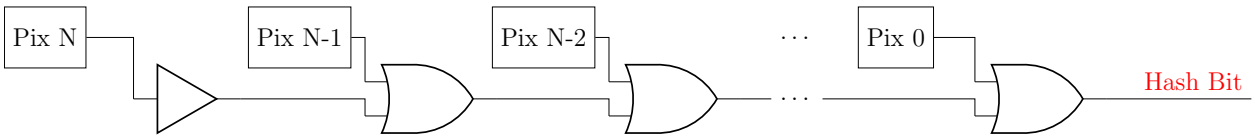


Figure 4.12: A representation of an OR chain used to generate a bit of the hash code.

This architecture is named **hash architecture** as the hash code produced is a hash code in the mathematical and computer science sense [124]–[126]. A hash function is commonly defined as "a mathematical function that maps element from a usually very large, or even infinite domain to a small range" [126], and the same definition is valid in this architecture. Hash functions are very common especially in the field of cryptography, where they are used to conceal the information via a key that is known only to the correct interlocutor, and security, where the evaluation of a hash code allows to verify whether data have been changed or not. In this architecture, the space of the possible values (hence, the domain of the hash function) is the space made of the possible states of the array. Considering that the matrix is $N \times N$ and that the information saved is binary (a pixel can either contain data or not), the set of possible states is made of 2^{N^2} elements, and a possible representation of an element is given by N^2-bits word. The hash code produced, that is the codomain of the hash function, is dependant on the number of projections that are used and equal to $2^{\mathcal{P}N}$, and a hash code can be represented by a $\mathcal{P}N$-bits word. The hash function used in this thesis lacks some of the properties important for the common hash functions; for example, many of the possible hash codes do not have a counterpart as a state of the physical array (more details in the next sections). The mathematical properties of the projections used are studied in Section 4.3.

Concerning the periphery of the detector, this architecture is slightly more complicated than the clockless column readout discussed before: now the hash code needs to be processed properly. In practise, there is a **decoder**, that for each hash code outputs one or more pixel addresses compatible with the given input. In case of multiple possible addresses found, the decoder orders them based on the *closeness* to one of the corners of the array: e.g., among all the possible hit coordinates, the encoder starts from the leftmost column among the row closest to the bottom. Each address is readout to verify whether it contains an hit or no: in case of affirmative result, the address is validated and moved toward the sensor output.

In case there are *too many* hits in the matrix, it may in fact happen that the addressed pixel does not contain an actual hit (this effect affects about 5% of addresses when there is around 1 hit every 200 pixels in the array, as detailed in Section 5.3). This is referred to as a **ghost address** through this thesis, and it is a consequence of employing a lossy compression algorithm. In such cases, the periphery logic discards the address and passes to the following one (more details in Section 5.1.5). As a result of this verification mechanism, neither fake hits nor missing hits are produced, simply the architecture consumes some extra power and time per readout cycle, i.e. increases its inefficiency.

Pixels are readout and checked for containing a real hit by a by-column, asynchronous ping mechanism, which uses a `SYNC` signal to time-align the pixel answer at the periphery, so that the readout cycle is unaffected by the pixel position in the column (which significantly alters the ping time delay). The fact that the addresses are just guessed (and not surely known) by the periphery drives the need for proper nomenclature, which is used through this chapter and the next one:

- **Tentative address**: a tentative address is an address created by looking at some hash words but not all of them. It can be valid or not. This concept is useful in the discussion on how the decoder works, in Section 5.1.5;
- **Valid address**: a valid address is an address calculated by the periphery that can mathematically lead to the hash code that is currently present in the device. It can be either correct or a ghost;
- **Correct address**: a correct address is an address of a pixel actually containing a hit;
- **Ghost address**: a ghost is a valid address that is not correct; i.e., an address that could be populated according to the hash code but actually is not.

Although it is implementation-dependent, for the case discussed in this work the readout pixel selection uses two signals: a vertical and a horizontal one that unequivocally select the pixel to be read. This limits the architecture true parallelism, as columns cannot be accessed independently in the pixel readout phase. A *sector* is therefore defined as the portion of the array that implements the hash architecture, and only one readout at a time must be performed in a sector; a sensor of practical size is planned to embody many sectors working in parallel.

Power considerations

Concerning the power consumption of the hash architecture, one of the main contribution comes from signals switching across "long" lines that traverse the entire matrix; in a standard flavour and with $\mathcal{P}$ projections, the long signal lines necessary to implement the proposed architecture are those listed in Table 4.3. Among the selection and synchronization signal, the *Hitmap* is the one responsible to communicate to the periphery the actual content of the pixel (i.e. hit or no hit); the name hitmap comes from the fact that a singe address may refer to, in actual implementations, more than a single physical pixel cell, to further optimize the readout; this solution has been indeed used in the RTL implementation of the hash architecture discussed in Chapter 5.

This number of long-lane transitions can be compared to the clockless column readout one listed in Table 4.1 or with the ALPIDE one. This comparison depends on both the number of projections and the size of the array. If a realistic number of projections (four) is set, capable of withstanding a good particle rate (see Section 5.3), within a square matrix having $N = 512$ pixels by side, the number of long-lane transitions is fixed[5] to 9 per readout for the hash architecture, to be compared with the average number of 6.5 for an ALPIDE-like square sensor of the same size.

	standard hash
Hash code creation	$\mathcal{P}$
Row selection	1
Read request	1
Hitmap communication	1
Sync	1
Reset request	1
Total	$\mathcal{P} + 5$

Table 4.3: Long-lane transitions in a standard flavour hash architecture with $\mathcal{P}$ projections.

The actual advantage in terms of power consumption is not apparent in this implementation as, by counting the number of transitions, the power consumption performance appears worse than ALPIDE one. This is due to a conservative implementation where priority was given to reliability and debug; nevertheless, in more stream-lined implementations, as is discussed in Section 5.4, this issue is tackled and architecture is far more promising. Furthermore, when also considering the power consumption due to the in-pixel readout logic, the hash architecture better compares with respect to the ALPIDE sensor.

Timing performance

For what concerns timing performance, this architecture is slower than the ALPIDE one, since address communication happens before actual data communication: one clock cycle is not enough. Moreover, the readout cycle needs to stay idle from the moment the readout is performed up to the moment the reset is carried out due to the impossibility of paralleling the readout by column. In the implementation discussed in Section 5.1, this means that at least three clock cycles need to pass before two subsequent readouts.

[5]This number is fixed in the sense that it does not depend on the size of the array. Nevertheless, when the array population gets higher, it is reduced because some of the hash bits are already high, but this is a second-order correction.

Many-hit Clogging

Many-hit clogging is a concept that must be addressed for this architecture: whenever there is more than one hit in the array, a ghost address may be reconstructed in the periphery. For example, if there are two hits in an architecture with two projections, it is not always possible to know the addresses of the pixels containing the data, due to the already discussed lossy algorithm employed. The issue is due to multi-hit collisions, a concept that was shown in Figure 2.4. Generalising this, the more hits there are in the array, the more ghost addresses are probable, and, therefore, the more time and energy are wasted trying to read empty addresses instead of correct ones. This issue cannot be removed, but can be tackled by adding more projections. The addition of a projection leads to an architecture that consumes more power, requires more gates, and is more crowded in terms of transistors and wires. In this work, the possibility of up to four different projections is explored. The performances of these devices are discussed and simulated in Chapter 5, where the issue of many-hit clogging is quantified.

4.3 Hash projections - a mathematical approach

This section mathematically addresses the problem of how to define, select, and use different hash projections; i.e., ways to create a hash code starting from a list of addresses in a pixel sensor. Here, as in previous sections, the sensor considered is a square $N \times N$ array. The most common and immediate projections are the so-called P_x and P_y projections: the column and the row of the address.

The fundamental concept is that each of these projections should convey as much information as possible, without overlapping with one another. A set of projections is ranked according to its **resolutive value**: the ability to obtain from the projected values the actual addresses that generated them.

In Section 4.3.1 the mathematical formalism is introduced together with a brief recall of some well-known properties of Diophantine equations, in Section 4.3.2 the discussion on how to understand which projections maximise the resolutive power is reported, in Section 4.3.3 a theorem showing how for low population fraction no ghost hits are produced is demonstrated, and in Section 4.3.4 two examples of optimal sets of projections in arrays with different size are given.

4.3.1 Formal framework

Definitions

Formally, the **address** of a pixel can be represented by the two-dimensional vector:

$$A = \begin{pmatrix} x_A \\ y_A \end{pmatrix} \qquad \text{with } x_A, y_A \in \mathbb{N},\ x \in [0, N[\wedge y \in [0, N[$$

where the position is identified by the x_A and y_A coordinate and N is the side of the square array. Consequently, a **projection**[6] is defined as a function in a discrete space as:

$$P : ([0, N[\times[0, N[) \rightarrow [0, N[\tag{4.3}$$

A projection applied to an address produces a scalar number, so they can be represented as a row vector[7]:

$$P_1 = (a_1, b_1) \qquad \text{with } a_1, b_1 \in \mathbb{Z}$$

In the most general case there is no constraint on a_1, b_1 to be either positive or smaller than N, but they have to be integers, as the array space is discretized. These numbers can be constrained without losing in generality, and this is shown through this section.

The projections are requested to map an $N \times N$ space to the interval $[0, N[$, hence the codomain of P_1 projection must be constrained when it is applied to a generic address. With $n \in \mathbb{Z}$, projections are applied as:

$$P_1 \cdot A = (a_1, b_1) \cdot \begin{pmatrix} x_A \\ y_A \end{pmatrix} = a_1 x_A + b_1 y_A + nN = \eta_{1A} \text{ with } \eta_{1A} \in [0, N[\tag{4.4}$$

and η always represents the value calculated applying a projection to an address. Projections application always includes this *module* operations consisting in adding or subtracting a multiple of the array size N. Through this section, the changes on the variable n is never tracked when integer values are added as this event is actually meaningless.

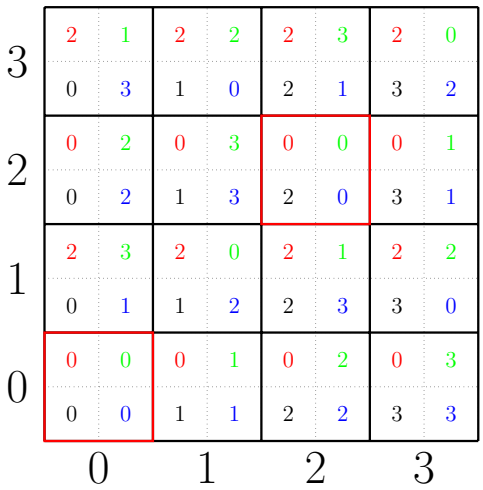


Figure 4.13: Example of projections in a $N = 4$ array. P_2 is uneven, the others are flat. P_0 and P_1 are optimal, just like P_0 and P_3. P_1 and P_3 are suboptimal because some pixels share the same pair of projected values (e.g. $(0, 0)$ and $(2, 2)$, in red, are both projected on $(0, 0)$).

[6]The word *projection* might be misleading as these are not projections in a geometrical sense: they map a bi-dimensional space into a one-dimensional one.

[7]This is a reduction to *linear projections*. In principle, also non-linear projections can be defined; in this work only linear projections are considered.

The value η_{1A} does not mean anything in *absolute terms*, but is informative only when compared to other elements (η_{1B} etc.). In Figure 4.13 some projections are represented in a square array. Intuitively some of these projections appear more useful than others for the purposes of the hash architecture; e.g., P_2 looks like the row projection P_Y but groups pixels in a suboptimal way. The projections characterisation and nomenclature (flat, uneven, and optimal) is formalised in the next section.

Another important concept is the connection between this mathematical framework and the actual application to pixel detectors: when a projection P_1 applied to the pixel A gives the value η_{1A}, and the same projection applied to another pixel B gives as a result the same number $\eta_{1B} = \eta_{1A}$, this means that pixel A and pixel B must be connected in the same chain of `OR` gates (like the one in Figure 4.12). Actually, this does not by any means imply that there is some sort of physical proximity between pixel A and pixel B, that can even be on opposite sides of the array.

Given these premises, the resolutive power of a set of projections lies both in the single projections themselves and in the relationship between them. Considering the projection as it is it can be:

- **Flat** if every element of the codomain appears exactly N times through the array (P_0, P_1 and P_3 in Figure 4.13);
- **Uneven** if there is at least one element of the codomain appearing a number of times different than N (P_2 in Figure 4.13).

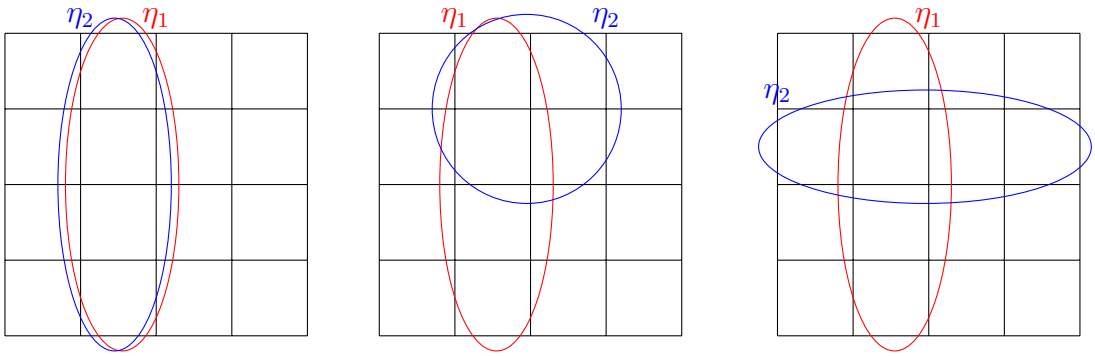
Naturally, *good* projections must be **flat** in order to maximise the resolutive power: if the value η_{1A} appears more times than the value η_{1B}, it will be harder to reconstruct an address belonging to the set of those that are projected to η_{1A}.

By comparing a generic projection P_i with another generic projection P_j, the pair of projections can be:

- **Trivial** if it is possible to create a bijective function f from the codomain of P_i to the codomain of P_j such that if the image of an address for the first projection is $P_i(A) = \eta_{iA}$ then the image of the same pixel for the second projection is $P_j(A) = \eta_{jA} = f(\eta_{iA})$ for every address A in the array.
- **Suboptimal** if there exists at least one pair of elements in the codomain of P_i and P_j, named η_{iA} and η_{jA}, which counterimage corresponds to more than one address in the array. If this is true for all the elements of the codomain, the two projections are also trivial; through this section, suboptimal always means suboptimal but not trivial.
- **Optimal** if for every pair of elements (η_{iA}, η_{jA}) in the codomain of the projections there is always one and only one corresponding element $A = (x_A, y_A)$ in the array.

A schematic representation of the relationship between two projections is shown in Figure 4.14. In the leftmost figure, projections are trivial as they superimpose, hence a second projection does not provide any further information. In the middle figure, projections are suboptimal as knowing the projected values it is not possible to reconstruct the pixel address (two addresses have the same projected values). In the rightmost figure, projections look optimal as every address is uniquely determined (they are optimal if this happens for every

possible pair of projected values).



(a) Trivial projections. **(b)** Suboptimal projections. **(c)** Optimal projections.

Figure 4.14: A representation of the relation between two projections. Here, circled in different colours, sets of pixels projected on the same value η_i by the two different projections. The actual value of η does not have any physical meaning.

In order to optimise the results with the given physical constraints (number of projections and dimension) it is best to use a set of **optimal** and **flat** projections[8], therefore an extensive search on how to find them has been carried out. Two trivial projections are physically the same projections (there is no actual difference in circuitry), while two suboptimal projections have less resolutive power as reconstruction is more likely to be wrong (these statements are demonstrated by using Monte Carlo simulations in Section 5.3).

Linear Diophantine equation resolution

In this section many **linear Diophantine equations** are shown: equations with integer values that ask for integer solutions. The results presented here can be found is several math books, like [127]. A linear Diophantine equation is an equation written like:

$$a \cdot \mathbf{x} + b \cdot \mathbf{y} = c \tag{4.5}$$

with integer parameters a and b, integer constant term c, and integer $\mathbf{x}, \mathbf{y}$ variables (through this section, for clarity, the variables of every equation are written in **bold**). It is a well known mathematical fact that this equation admits solution if and only if the constant parameter c is a multiple of the greatest common divisor of the parameters a and b. In particular, if $c = km$, where m is the greatest common divisor of a and b, and (x_0, y_0) is a particular solution, then $(x_0 + k\frac{b}{m}, y_0 - k\frac{a}{m})$ with $k \in \mathbb{Z}$ exhausts the set of infinite possible solutions.

Often, the concept of **coprime** is used as well: two (or more) natural numbers are coprime when their greatest common divisor is one[9]. When a set contains some zeros, then

[8] A set of optimal projections is a set of projections pairwise optimal.

[9] For example, the numbers 8 and 9 are coprime, despite neither of them is a prime number, since 1 is their only common divisor. On the other hand, 6 and 9 are not coprime as they are both divisible by 3. The numerator and denominator of a reduced fraction are coprime, by definition.

these numbers must not be accounted for in the coprime definition, and a set made of one number only is never a set of coprimes.

4.3.2 Projections scoring

Projections flatness

A projection is flat if every element of the codomain appears exactly N times through the array. The first part of this demonstration shows that every value in $[0, N[$ is populated at least once. By applying a generic projection with parameters a and b to a generic address with coordinate $\mathbf{x}$ and $\mathbf{y}$ the result is:

$$a \cdot \mathbf{x} + b \cdot \mathbf{y} + N \cdot \mathbf{n} = \eta \tag{4.6}$$

that must be solvable in the variables $\mathbf{x}$, $\mathbf{y}$, $\mathbf{n}$. The constant term η is the value calculated by the projection, N is the array size, and $\mathbf{n}$ is the variable added to renormalise the outcome to the interval $[0, N[$. Equation 4.6 is a three variable Diophantine equation and, like the two variable case, it can be shown to admit solutions if and only if the constant parameter η is a multiple of the greatest common divisor of the parameters a, b and N.

This equation must have at least one solution for every η from $\eta = 0$ up to $\eta = N - 1$. This happens if the only common divisor of a, b and N is one, i.e. $\{a, b, N\}$ must be a set of **coprimes**. This condition is taken as verified in the following calculations: if it is not true, then some elements in the codomain do not appear; i.e., the projection is not flat but uneven[10].

Now, the second part of the demonstration shows that this condition also forces the number of solutions in the domain to be exactly N. There are two different possibilities that are discussed, as the set $\{a, N\}$ can either be a set of coprimes or not (the same demonstration can be carried out by using $\{b, N\}$).

- If a and N are **coprime**, a fixed row and a fixed projected value can be considered; i.e., $y = \tilde{y}$ and $\eta = \tilde{\eta}$ are set, and there is a search for pixels in this particular row projecting on this particular value. This translates into solving the equation

$$a \cdot \mathbf{x} + N \cdot \mathbf{n} = \tilde{\eta} - b\tilde{y} \tag{4.7}$$

  In this case, as a and N are coprime, the equation admits infinite solutions. If $S_1 = (\bar{x}, \bar{n})$ is a particular solution, as discussed in Section 4.3.1, the general solution can be written as $S = (\bar{x} + kN, \bar{n} - ka)$, with $k \in \mathbb{Z}$. The navigation in the solution space is independent from both $\tilde{y}$ and $\tilde{\eta}$, that only select the particular solution. This means that this Diophantine equation admits only one solution in the x domain: for every specific row there is exactly one pixel projecting to a specific value. This translates into the fact that for every row, all values are populated exactly once. By considering

[10] By looking back at Figure 4.13, P_2 is uneven as 2 and 4 are not coprime and actually some values (1 and 3) are never populated.

that there are N rows, it is possible to conclude that there actually are N values for every η, hence the projection is flat.

- If a and N are **not coprime**, this means that there exist a common divisor e and two more coprime integers f, g such that:

$$a = ef \qquad N = eg \tag{4.8}$$

applying the projection to a generic pixel fixing the projected value $\eta = \tilde{\eta}$, the equation is:

$$ef \cdot \mathbf{x} + eg \cdot \mathbf{n} + b\mathbf{y} = \tilde{\eta} \tag{4.9}$$

First, it is necessary to count the number of rows that would admit solution considering y as a parameter: rewriting Equation 4.9 as:

$$ef \cdot \mathbf{x} + eg \cdot \mathbf{n} = -by + \tilde{\eta}$$

and considering that f and g are coprime, this equation is solvable for every $\tilde{\eta}$ only if it is possible to make the term e appear on the right hand side. Hence, it is solvable for a specific row y if it is possible to write

$$\tilde{\eta} - b \cdot \mathbf{y} = e \cdot \mathbf{h} \tag{4.10}$$

for some integer h. This is a linear Diophantine equation as well in the variable $\mathbf{y}$ and $\mathbf{h}$ always solvable as the parameters b and e coprime (else the set $\{a, b, N\}$ would not be a set of coprimes). Once a particular solution $S_1 = (\bar{y}, \bar{h})$ is found, the general solution can be written as

$$S = (\bar{y} + ke, \bar{h} - kb) \tag{4.11}$$

with $k \in \mathbb{Z}$. This means that navigating through the solutions space there are exactly $N/e = g$ solutions in the y domain. This demonstration shows that in this case there are, for every possible fixed value $\tilde{\eta}$, exactly g rows where this projected value appears. Now it is necessary to count how often such a value actually appears in that row. By substituting Equation 4.10 in Equation 4.9 and simplifying the common terms, the equation obtained is:

$$f \cdot \mathbf{x} + g \cdot \mathbf{n} = h \tag{4.12}$$

in the variables $\mathbf{x}$ and $\mathbf{n}$, always solvable as f and g are coprimes. By navigating once again the solution space, starting from a particular solution, the result is:

$$S = (\bar{x} + kg, \bar{n} - kf) \tag{4.13}$$

with $k \in \mathbb{Z}$ and there are exactly $N/g = e$ valid solutions within the x domain.
This demonstration shows that, in this case, for every fixed $\eta = \tilde{\eta}$, there are g rows that contain each e addresses that project on $\tilde{\eta}$. Hence, through the array, this projected value appears $e \cdot g = N$ times, and the projection is flat.

To summarise the whole demonstration, **a projection is flat if and only if $\{\mathbf{a}, \mathbf{b}, \mathbf{N}\}$ is a set of coprimes**.

Projections triviality

Projections triviality (defined in Section 4.3.1) is defined for a pair of projections. Consider the following:

$$P_1 = (a, b) \qquad P_2 = (c, d) \tag{4.14}$$

First, a **null parameter** is defined as a parameter of a projection that is either zero of a multiple of N (it is easy to understand that these two situation are analogous). The case of projections with at least one null parameter are discussed separately.

- If a projection has both its parameters null, it is uneven, brings no information and cannot be used for the hash architecture purposes in any way.
- If the two projections are both flat and have a null parameter in the same position, they are trivial as they give the same information (either address row or column).
- If one projection has a null parameter while the second projection has valid parameter on that position, then the two projections are non-trivial. This happens because one projection is projecting on rows (or columns) whereas the second one is projecting either on diagonals or on columns (or rows). This demonstration is straightforward and just like the other ones presented in this section, and is not reported here.
- If all parameters are non null, this means that both projections project on diagonals. The triviality condition, for any couple of pixels A and B, can be written as:

  $$P_1(A) = P_1(B) \iff P_2(A) = P_2(B)$$

  It is possible to verify both conditions by putting these two statements in a system of equations. By applying the two projections as in Equation 4.4:

  $$\begin{cases} a \cdot \mathbf{x}_A + b \cdot \mathbf{y}_A = a \cdot \mathbf{x}_B + b \cdot \mathbf{y}_B + N \cdot \mathbf{n} \\ c \cdot \mathbf{x}_A + d \cdot \mathbf{y}_A = c \cdot \mathbf{x}_B + d \cdot \mathbf{y}_B + N \cdot \mathbf{m} \end{cases} \qquad \begin{cases} (cb - da) \cdot \mathbf{x}_A = (cb - da) \cdot \mathbf{x}_B + N \cdot \mathbf{n} \\ (cb - da) \cdot \mathbf{y}_A = (cb - da) \cdot \mathbf{y}_B + N \cdot \mathbf{m} \end{cases}$$

  The two projections are trivial if and only if this system is true for every $\mathbf{x}$ and $\mathbf{y}$ in the array. Therefore, two diagonal projections are trivial if and only if

  $$|cb - da| = \mathbf{k}N \tag{4.15}$$

  with $\mathbf{k} \in \mathbb{Z}$.

In particular, by comparing the two projections:

$$P_1 = (a, b) \qquad P_2 = (a + iN, b + jN) \tag{4.16}$$

with $i, j \in \mathbb{Z}$, these projections are always trivial. This helps constraining the diagonal projections parameters without losing in generality.

Projections parameters

In the last two sections, some constraints on projections have been put in order for them to have a good resolutive power. Taking advantage of the obtained results, before passing to the study of pairwise optimal projections, it is possible to say that:

- $0 \leq a \leq N-1$ and $0 \leq b \leq N-1$ can be constrained without losing generality, as the other projections are a trivial version of one of the set.
- $\{a, b, N\}$ must be a set of coprimes otherwise the projection is uneven.
- Many diagonal projections P_{ba} and P_{cd} are trivial one another, and that can be checked by looking if Equation 4.15 admits integer solutions for the variable $\mathbf{k}$.

Starting from these facts, to ease the notation and formal description, the projections used in the next paragraphs are:

- P_x is the projection that looks only at the x value of the pixel. In particular:

$$P_x = (1, 0) \tag{4.17}$$

  All the other projections having a zero in the second position are either uneven or trivial with respect to P_x, as can be easily seen.
- P_y is the projection that looks only at the y value of the pixel. In particular:

$$P_y = (0, 1) \tag{4.18}$$

  All the other projections having a zero in the first position are either uneven or trivial with respect to P_y, as can be easily seen.
- P_{ab} is the diagonal projection with both a and b in $]0, N[$. In particular:

$$P_{ab} = (a, b) \tag{4.19}$$

  These projections are flat if and only if $\{a, b, N\}$ is a set of coprimes.

These three sets of projections describe all the possibilities.

Optimal projections

Consider two projections P_1 and P_2 defined as in Equation 4.14. By applying the same projections to two generic addresses A and B:

$$\begin{aligned}
\eta_{1A} &= P_1 \cdot A = a \cdot \mathbf{x}_A + b \cdot \mathbf{y}_A + N \cdot \mathbf{n} \\
\eta_{1B} &= P_1 \cdot B = a \cdot \mathbf{x}_B + b \cdot \mathbf{y}_B + N \cdot \mathbf{n}' \\
\eta_{2A} &= P_2 \cdot A = c \cdot \mathbf{x}_A + d \cdot \mathbf{y}_A + N \cdot \mathbf{m} \\
\eta_{2B} &= P_2 \cdot B = c \cdot \mathbf{x}_B + d \cdot \mathbf{y}_B + N \cdot \mathbf{m}'
\end{aligned}$$

If the two projections are optimal, this means that:

$$\begin{cases} \eta_{1A} = \eta_{1B} \\ \eta_{2A} = \eta_{2B} \end{cases} \implies A = B \tag{4.20}$$

In suboptimal and trivial pairs of projections this is not true. To further bring on calculations on Equation 4.20 it is needed to discuss the cases in which some parameters are null, using the notation introduced in the last section.

The first study reported here shows whether the pair made by P_x and P_y is optimal or not. From a physical perspective, this is clear, as having a pair (x, y) there is only one address in the array capable of producing these values. By applying the projections definition to the system in Equation 4.20:

$$\begin{cases} \mathbf{x}_A = \mathbf{x}_B + N\mathbf{n} \\ \mathbf{y}_A = \mathbf{y}_B + N\mathbf{m} \end{cases}$$

Considering that all x and y must be in $[0, N[$ according to the constraints that were set on the domain, this forces $n = m = 0$, and there is only one solution where this system of equations is satisfied: when $A = B$ and A and B are actually the same pixel. This is exactly the definition of optimality: the projected values are the same if and only if the pixel is the same; hence, these two projections are optimal.

As a following step, the optimality between two projections belonging to the set P_{ab} is studied, in particular P_{ab} and P_{cd}. By using once again Equation 4.20:

$$\begin{cases} a \cdot \mathbf{x}_A + b \cdot \mathbf{y}_A = a \cdot \mathbf{x}_B + b \cdot \mathbf{y}_B + N \cdot \mathbf{n} \\ c \cdot \mathbf{x}_A + d \cdot \mathbf{y}_A = c \cdot \mathbf{x}_B + d \cdot \mathbf{y}_B + N \cdot \mathbf{m} \end{cases} \rightarrow \begin{cases} (cb - da) \cdot (\mathbf{x_A} - \mathbf{x_B}) = N \cdot \mathbf{n} \\ (cb - da) \cdot (\mathbf{y_A} - \mathbf{y_B}) = N \cdot \mathbf{m} \end{cases} \tag{4.21}$$

where this operation is possible as all the parameters are different from zero. The topmost one is discussed, but everything can be applied, *mutatis mutandis*, to the other one. This Diophantine homogeneous equation in the variables $(\mathbf{x_A} - \mathbf{x_B}, \mathbf{n})$ always accepts infinite solutions, and a trivial one is $S_1 = (0, 0)$. To find the generic solutions starting from the trivial one it is necessary to discuss the value of the greatest common divisor between the two parameters N and $|cb - da|$.

- If these two numbers are coprime, with $k \in \mathbb{Z}$, the set of all possible solutions can be written as:

  $$S = (kN, -k(cb - da))$$

  Looking at the solutions with $|k| \geq 1$ in this equation, x_B goes outside of its domain. Therefore, the trivial solution is the only one within the domain and the pixel must be the same $x_A = x_B \wedge y_A = y_B$. **The projections are optimal**.
- If $|cb - da|$ and N are not coprime, that means that their common divisor e is higher than 1, i.e.:

  $$cb - da = fe \qquad N = ge$$

  with f and g integers. In this case, solutions to Equation 4.21 are written as:

  $$S = (kg, kf)$$

  This means that more than one solution is possible still keeping x and y both $\leq N$, as $g < N$; hence at least two pixels give the same pair of projected values under the two projections and, therefore, **the projections pair is suboptimal**.

As a conclusion, **two diagonal projections $\mathbf{P_{ab}}$ and $\mathbf{P_{cd}}$ are optimal if and only if $\mathbf{|cb - da|}$ and $\mathbf{N}$ are coprime**[11]. This also demonstrates the fact that trivial projections are a particular case of suboptimal projections.

[11] By looking back at Figure 4.13, P_1 and P_3 are suboptimal as $|cb - da| = 2$, not coprime with $N = 4$.

As a last case, optimality between a generic diagonal projection P_{ab} and a projection with a null parameter, like P_x, must be studied. The equation to solve is Equation 4.20:

$$\begin{cases} a \cdot \mathbf{x}_A + b \cdot \mathbf{y}_A = a \cdot \mathbf{x}_B + b \cdot \mathbf{y}_B + N \cdot \mathbf{n} \\ \mathbf{x}_A = \mathbf{x}_B + N \cdot \mathbf{m} \end{cases} = \begin{cases} b \cdot (\mathbf{y_A} - \mathbf{y_B}) = N \cdot \mathbf{n} \\ \mathbf{x}_A = \mathbf{x}_B + N \cdot \mathbf{m} \end{cases} \tag{4.22}$$

and, by repeating the very same argument already proposed for last calculations, the two projections are optimal if and only if b and N are coprime.

Repeating calculations with P_y:

$$\begin{cases} a \cdot \mathbf{x}_A + b \cdot \mathbf{y}_A = a \cdot \mathbf{x}_B + b \cdot \mathbf{y}_B + N \cdot \mathbf{n} \\ \mathbf{y}_A = \mathbf{y}_B + N \cdot \mathbf{m} \end{cases} = \begin{cases} a \cdot (\mathbf{x_A} - \mathbf{x_B}) = N \cdot \mathbf{n} \\ \mathbf{y}_A = \mathbf{y}_B + N \cdot \mathbf{m} \end{cases} \tag{4.23}$$

and the two projections are optimal if and only if a and N are coprime.

To summarise everything demonstrated in these few pages:

- P_x and P_y are always flat and optimal one with the other. All other projections with null parameters are either trivial with respect to these two or uneven.
- a diagonal projection P_{ab} is flat if and only if $\{a, b, N\}$ is a set of coprimes.
- a diagonal projection P_{ab} is optimal with respect to P_x if and only if a and N are coprime, optimal with respect to P_y if and only if b and N are coprime.
- Two diagonal projections P_{ab} and P_{cd} are optimal if and only if $\{|cb - da|, N\}$ is a set of coprimes.

4.3.3 Number of perfectly reconstructable addresses

It is possible to show that **if an array that implements M optimal projections, if the periphery sees M reconstructed addresses, none of them can be a ghost**.

The demonstration is performed by *reductio ad absurdum*: suppose that there is a ghost address X when M addresses have been reconstructed. This address X must share each one of its projections with a valid address (the definition of a ghost). Now assume that, by some ordering, there is only another pixel A sharing the same projection by P_1, i.e. $P_1(X) = P_1(A)$. Then, looking at P_2, the projected value cannot be shared with A, as this would go against the hypothesis that projections are optimal; i.e., the pair of projections P_1 and P_2 cannot distinguish between A and X. Hence, it must be shared by another address B. By iterating this process, the $(M-1)$th projection must be shared with the $(M-1)$th pixel. Now, finding a pixel sharing the Mth projection, such a pixel cannot be any of the pixels that already share another projection, as this would go against the optimality hypothesis. But the pixels set has actually been exhaused. Therefore, such a ghost address cannot exist. This argument remains valid if more than one pixel shares a given projection (the set gets exhausted earlier), or if the existence of a ghost when there are less than M valid addresses is studied (the set is smaller). The conclusion is that if there are M optimal projections implemented, there cannot be ghosts whenever there are no more than M reconstructions.

For example, with two optimal projections there cannot be a ghost if exactly two addresses have been reconstructed, with three optimal projections there cannot be a ghost when three addresses have been reconstructed and so on.

A corollary to this theorem, actually less general than the one just demonstrated, is that **when there are M − 1 hits in an array implementing M optimal projections, there cannot be any ghost address**.

4.3.4 Examples of optimal projections sets

Even arrays

The first example is a simple case, quite common in digital electronics: an **even-sized** square array with size:

$$N = 2^n$$

where, for MAPS, n is usually between 7 and 11 [54], [128]. The even-sized condition is actually enough to constrain the maximum number of optimal projections that can be used at the same time[12].

First, it is useful to discuss for an even-sized array which diagonal projections are optimal with respect to P_x or P_y, and which pairs of diagonal projections are optimal. The results are summarised in Table 4.4 where the four parameters defining the two projections are discussed as either even or odd cases. It is important to note that the diagonal on the rightmost table is a diagonal of †: two projections with the same properties in term of parameters evenness are never optimal.

	a_{even}		a_{odd}	
	b_{even}	b_{odd}	b_{even}	b_{odd}
P_x	X	†	ok	ok
P_y	X	ok	†	ok

		a_{even}		a_{odd}	
		b_{even}	b_{odd}	b_{even}	b_{odd}
c_{even}	d_{even}	X	X	X	X
	d_{odd}	X	†	ok	ok
c_{odd}	d_{even}	X	ok	†	ok
	d_{odd}	X	ok	ok	†

Table 4.4: Optimality between diagonal projections and null-parameter projections and among each other in an even array. † means these parameters make the pair suboptimal, X means that one of the two projections is uneven, ok means that the pair is optimal.

From this table, it is possible to see a match between some projections with a null parameter and some diagonal projections in terms of optimality: both P_x and P_{eo}, i.e.,

[12]The power of two constraint is looser than the even-size constraint such as for N = 6: some set of projections that are optimal in the case of an array with a power of two size are suboptimal in even matrices, where the size might admit more divisors

a diagonal projection with a even and b odd, are optimal and suboptimal with the same projections; the same holds for P_y and P_{oe}. This can be seen by comparing the two tables reported: the smallest one on the left hand size is a repetition of the middle two rows of the table on the right hand side.

In order to create an optimal projections set, it is possible to arbitrarily start by using the two simplest projections P_x and P_y; according to the demonstrations shown in the last section, these two are optimal (it is possible to start with a P_{oe} and a P_{eo} according to the previous argument, everything would remain the same mathematically). Adding a third projection, it has to be a diagonal one, of the type P_{ab}. To be flat and optimal with respect to the two projections already used, it is necessary for both a and b to be odd, hence it must be a P_{oo}. A fourth projection would still need to be a P_{oo}, but this would be suboptimal with respect to the one just added to the set. Hence, it is impossible to have a fourth projection starting from P_x and P_y. The largest optimal set of projections is:

$$\{P_x, P_y, P_{ab}\} \qquad \text{with } a, b \text{ odd} \tag{4.24}$$

Alternative sets can be obtained by using more diagonal projections, but never more than three. In Figure 4.15 a representation of the possible projections that can be used for an optimal set in an even array is given.

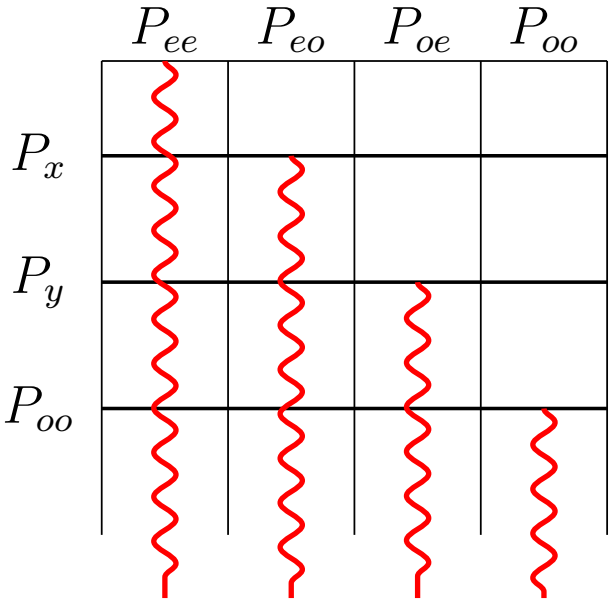


Figure 4.15: A representation of the creation of a set of optimal projections in an even-sized array. When the first projection is chosen (top left), some diagonal projections become unavailable (and are hence stricken out by the red line). The process continues going top down. After three optimal projections are added to the set, there is no other projections that can be added maintaining optimality. P_{ee} is uneven, hence never usable.

Prime matrices

As an alternative on the other side of the spectrum, it is possible to consider an array $N \times N$ with N prime. In this kind of array, all diagonal projections are flat, because the set $\{a, b, N\}$ is a set of coprimes independently from the values of a and b (when they are not both null). All diagonal projections are also optimal with respect to P_x and P_y projections for the same reason. By looking at optimality between diagonal projections it is possible to conclude that **no suboptimal pairs of projections can exist**, only optimal or trivial. For example, for $N = 5$, the following set of projections can be used:

$$\begin{aligned} P_x &= (1, 0) \\ P_y &= (0, 1) \\ P_{11} &= (1, 1) \\ P_{12} &= (1, 2) \\ P_{13} &= (1, 3) \\ P_{14} &= (1, 4) \end{aligned} \tag{4.25}$$

All other projections that can be define would be trivial with respect to one of this set: for example P_{21} is a trivial version of P_{13} ($|cb - da|$ in this case is 5, a multiple of the matrix size that is 5 as well, hence the two projections are trivial according to equation 4.15).

It is clear how this approach, brought to the extreme, is a loss of the actual goal of the architecture: the array here is made of 25 pixels, and to try and reconstruct each address several quite complicated connections are implemented, and a hash code of 30 bits (5 bits per each hash word, 6 hash words) is generated when 25 bits (one-hot encoded; a framing architecture) would have been enough. Nevertheless, this set of projections is mathematically guaranteed to perfectly reconstruct only 6 addresses at a time, whereas the one-hot encoding always performs perfect reconstructions.

Chapter 5

Hash architecture implementation and performance

This chapter deals with the actual implementation of the hash architecture, presents the Monte Carlo simulations performed, and discusses the performance of this architecture in comparison with the current state-of-art. In Section 5.1 the physical design implementation is discussed; in Section 5.2 a more formal analysis of the power and timing performance of the architecture is presented from an analytical point of view; in Section 5.3 the Monte Carlo simulations used to stress the architecture and their output; in Section 5.4 some further improvements to the implementation discussed are outlined.

5.1 Physical implementation

The hash architecture has been introduced from its defining principles in Chapter 4: it is thereafter shown an actual RTL implementation with realistic constraints. The work has been organised in a standard hierarchical way: the different simple modules combine to form the schematic of the whole array. These modules are presented together with some schematics (where relevant to show some of the functions implemented). Whereas everything discussed so far was abstract and valid for any implementation, from now on the technological node is fixed as LFoundry 110 nm, as was shown to be valid for the ARCADIA sensor. The cells used, whereas they may be optimised for some figures of merit (such as the speed or the power consumption) are chosen as the best trade-off among the various possibilities.

Within the context of the hash architecture development, different flavours have been theorised. The mathematical approach pursued in the previous chapter (Section 4.3) shows how to find and rate the best sets of hash projections, in a general context valid for any size. In the next Section 5.3, presenting the Monte Carlo simulations, different sizes, aspect ratios, and number of projections are explored, while the RTL implementation focuses only on a particular version of one of these architectures. The first choice concerns the size of the device. Here, a square device made of 512×512 pixels is discussed. It is divided into

rectangular regions (higher than wide), each implementing its own hashing architecture, thus improving the readout parallelism. Every independent rectangular region (called a section or a sector), in a similar way to the ARCADIA-MD1 architecture discussed in Section 3.1, has an aspect ratio of 16:1 and is made of 32×512 pixels. The sections implement a **clustering architecture**, where the addressable unit is a square-shaped super-pixel (called pixel region) made up of 4 pixels (clustering architectures were represented in Figure 4.4b). This means that a section has a total of 16,384 pixels and 4,096 super-pixels. This subset of the array is actually a rearrangement of a square array with a hash side of 64 rearranged as 16×256. From the readout point of view, sectors are completely independent entities that are not communicating with each other; there is no signal crossing horizontally a section border. Pixels have one buffer each, therefore, after one hit, the pixel is full and cannot process more data. A pixel region containing at least one full pixel immediately starts broadcasting the corresponding hash bits, not waiting for any external trigger. Sixteen different sections are aligned horizontally to form the array. A scheme of this implementation can be seen in Figure 5.1.

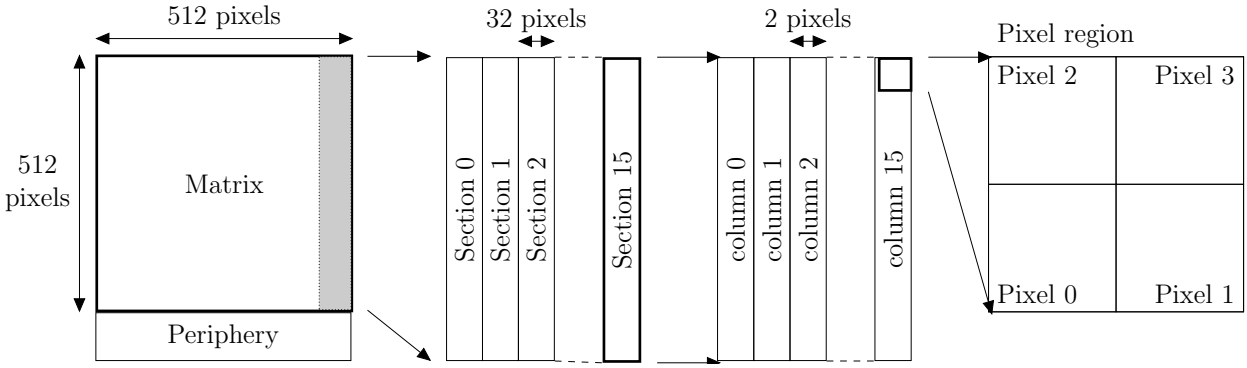


Figure 5.1: A representation of the array development discussed in this section. Although columns and pixel regions appear to be replicated, each replica is actually customised accordingly to the position in the array to implement the correct hashing connections. More details are discussed through this section.

Concerning the actual sensor size, the same size of ARCADIA-MD1 was taken as a reference; hence, pixels are square-shaped with a side of 25 µm (hence pixel regions are squares with a side of 50 µm). In this implementation, the entire array is 12.8×12.8 mm^2.

The second choice defines the set of projections embodying the hash function; the following four projections are used (in Section 5.1.4 more details on how to implement these projections on a rectangular-shaped array):

$$\begin{aligned} P_X &= (1, 0) \\ P_Y &= (0, 1) \\ P_U &= (1, 1) \\ P_V &= (1, -1) \end{aligned} \tag{5.1}$$

From a mathematical point of view, after the discussion of Section 4.3 and considering that the chosen rectangular sector is mathematically equivalent to a square array with $N = 64$,

this set of projections cannot be optimal. The pair P_X and P_Y is optimal, and by including either the P_U or the P_V projections the set remains optimal; however, the set of all four projections is suboptimal, as P_U and P_V are a suboptimal pair of projections. All projections are flat.

The third step requires implementing a proper readout routine. Here, a very safe option is chosen to build an architecture having good performance in terms of fake hits and lost hits, with the trade-off of worse power consumption and lower maximum hit rate. The implementation discussed here must be taken as a tentative for a very first flavour of the hash architecture: alternatives taking advantage of all the peculiarities of the hashing scheme might be implemented in future versions, after this flavour is ported to silicon and confirms the expected performances. Details on the hit readout are in Section 5.1.1; three signals are used to address the proper pixel region, and the answer is the pixel region hitmap with one-bit overhead.

5.1.1 Pixel region

Although the smallest addressable building block of an array is a pixel, its functioning depends on the **silicon design** and the **analogue architecture**. The domain of the digital architecture, in fact, starts after the digital signal is produced inside the array and the output of a voltage comparator goes high.

From a digital point of view, the tasks of a pixel region are the following:

- buffer the hit when a signal is created by the analogue architecture;
- trigger the hash network whenever at least one pixel contains data;
- broadcast the test pulses when requested;
- save and implement pixels configuration;
- implement correct readout-and-reset operations.

In the current implementation, the pixel region is very simple (the majority of transistors are actually used to implement the configuration logic, which was not optimized). The pixel digital logic is identical for every pixel region, with the exception of few changes on the wiring of five signals, all realized by enabling/disabling vias, is described below. Concerning the **size** of the pixel region, a reasonable implementation of the has architecture in-region logic (with good power and timing performances) discussed here requires a total area (spread over 4 physical pixels) of about

$$500\,\mu m^2 \tag{5.2}$$

where, not considering the configuration logic, the size shrinks to about

$$250\,\mu m^2 \tag{5.3}$$

These numbers are compared with the current state-of-the-art in Chapter 6.

Hits buffering

In the RTL implementation here discussed there is one memory bit (implemented as a latch to optimise area and power) for each pixel; hence, a total of four bits of memory per pixel region. Considering that the clock is not propagated towards the array, synchronous memories cannot be used, and the output of the comparator must be latched. The type of latch used in this implementation is chosen looking at its size, the delay, and the power consumption.

This memory can be reset via a proper signal, namely its clear signal. Clear has priority over the set. Considering that the logic implemented is not safe for superimposing set and reset (collisions are possible if a hit is latched while the digital architecture is trying to reset the pixel), **S**et-**R**eset (SR) latches cannot be used, and safer options are preferred.

Configuration of the pixel

The pixel region configuration mechanism has been inspired by the work performed for ARCADIA-MD1, nevertheless the actual implementation is slightly different. A single pixel can be configured at two different levels: **globally** and **locally**. Local configuration bits are stored inside the pixel region in proper latches. These latches can be SR latches to reduce the number of transistors, as the configuration set-up is performed only by the periphery and is collision safe. Global configuration bits, on the other hand, are saved in the periphery and run through a whole column; they are therefore shared by all pixel regions aligned vertically, making it impossible to configure differently pixels belonging to pixel regions in the same column. Although all configuration bits can be set up globally, only some of them have a local counterpart (due to spatial limitations).

There are four configuration bits:

- Injection Enable (local): if set to one, the pixel is injectable and will respond to test pulses accordingly;
- Mask (local): if set to one, the pixel is masked. In a masked pixel the internal latch is not updated when the compartor output is high, nor when an injection is requested.
- Injection Digital (global): an incoming injection will be digital; i.e., the injection signal will be directly routed to the in-pixel buffer. If this bit is set to 0, the injection will be connected to the pixel front-end and test the full analogue readout chain;
- Bypass (global): whenever bypass is set, the local configuration bits are not used, but values are read from global lines used for the configuration in their place.

In Figure 5.2 it is possible to see how both configuration lines are implemented before hit buffering. This means that if, for example, masking is set, then the buffer is kept empty. In Figure 5.3 it is possible to see how the bypass signal is used to overwrite a configuration signal (in this case the mask): if both the bypass signal is one and the `global mask` signal is zero, then the mask is always removed (and therefore the pixel is receptive); in the other cases, the value is read from the register inside the pixel. This feature is added to help in cases where some issues during chip production may cause impediments for the correct

functioning of the device.

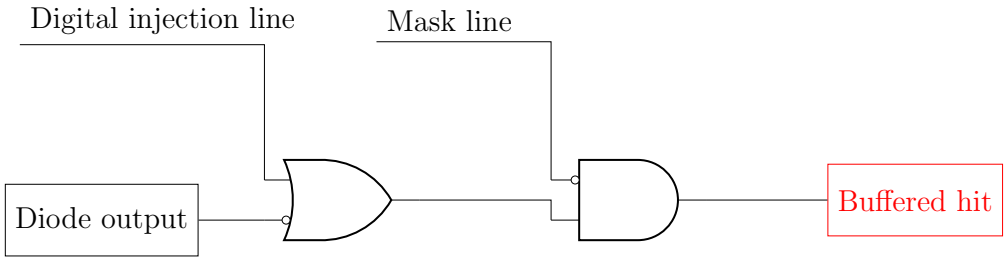


Figure 5.2: A simple representation of how digital injection and masking are implemented for a single pixel in a pixel region.

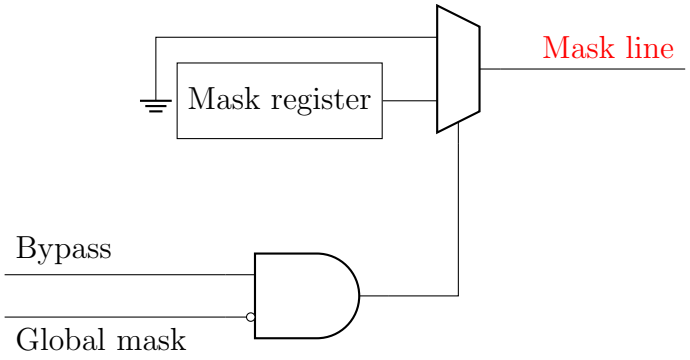


Figure 5.3: A representation of how global signals can be used to overwrite the value of local configuration registers. In this case, mask register is shown. When Bypass is high and global mask is low, the mask line is hard wired to zero.

In both these two schematics, that illustrate only how configuration affects the behaviour of the chip and not how to change the configuration of the device, only what is relative to one pixel is shown. This architecture is repeated for the four pixels that make up the pixel region. When adding more pixels, optimisation procedures reduce the number of transistors: the schematics shown here are used to show the functionality and do not represent the actual gates inside a pixel region.

Test pulse injection

As anticipated, test pulses can be injected both on the analogue and on the digital side. When injection is analogue, the pulse is circulated as a signal to the `external tp` port of the analogue front end and is propagated through the whole chain; when injection is digital, it is buffered directly inside the pixel latch, therefore skipping the analogue front-end.

The test pulse is propagated in a single vertical line common to all pixel regions inside a column, and only the pixels with injection enable bit high respond accordingly. This is the reason why it is important to have a local bit for injection enabling. This approach makes it easy to inject complicated patterns inside the array for different tests, such as the threshold calibration.

HASH code modification

For what concerns the address decoding, the hash function transitions to high if at least one of the pixels of the region contains data. This means that each OR gate that makes up the chain represented in Figure 4.12 is preceded by a 4OR, the input of which are the four pixel latches. Hash signals are properly connected by implementing four different projections, and a detailed discussion on how to properly connect these signals is in the dedicated Section 5.1.4.

Readout-and-reset

Concerning readout-and-reset, the periphery uses a total of three signals to select and operate on the pixel: READ, CLEAR, and ROWSEL. The first two travel vertically through the column, whereas the latter travels horizontally through the row. The pixel region responds to readout-and-reset routine by setting its internal hitmap on a bus named PID (**P**attern **ID**) and by raising a line named SYNC used to tell when the pattern is valid. Both these signals travel vertically on lines shared among the whole column; therefore, it is important not to have multiple pixel regions trying to communicate at the same time in a given sector.

A pixel region is selected both for communication and for reset when the horizontal signal and a vertical one meet. The signal ROWSEL actually enters two AND gates that are used to select whether this pixel should broadcast its information or not. READ and CLEAR are both two clock periods wide and arrive separated by one clock cycle. The usage of two signals that overlap partly identifies three time regions, separated by exactly one clock cycle: READ high with CLEAR low, both high, READ low with CLEAR high. These are used, respectively, to broadcast the PID signal, broadcast the SYNC signal, and perform internal reset of the pixel region. If there is only the horizontal signal high, or there is a vertical signal high without a horizontal counterpart, the pixel region does not perform any action, as it is not the selected one. There is no triggering on any edge of a signal; this approach is chosen for this hash *safe flavour* as it makes timing closure easier.

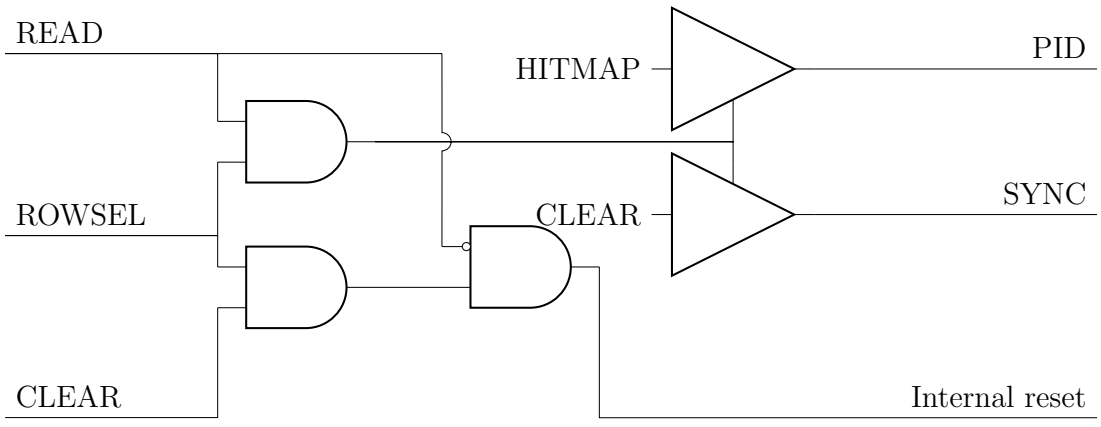


Figure 5.4: A schematic of how the pixel region module implements readout-and-reset.

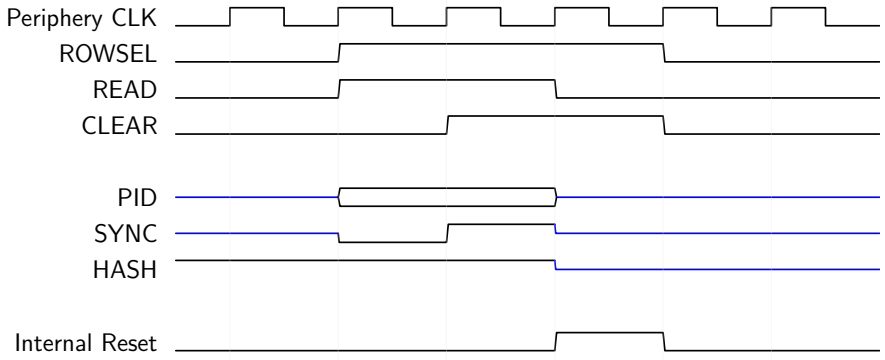


Figure 5.5: A timing table for a super pixel readout-and-reset. This table has been confirmed via RTL simulations, and for simplicity it is not representing the delays due to signals travel speed (such delays were included in all the simulations discussed later).

A schematic of the pixel region readout-and-reset logic can be seen in Figure 5.4, while in Figure 5.5 the timing table of the signals used for the readout-and-reset routine are shown, together with the internal reset. In Figure 5.6, in detail, a schematic taken from Synopsys Design Compiler [129] where the readout routine has been synthesised stripping the configuration and the hash propagation from the pixel region (for readability).

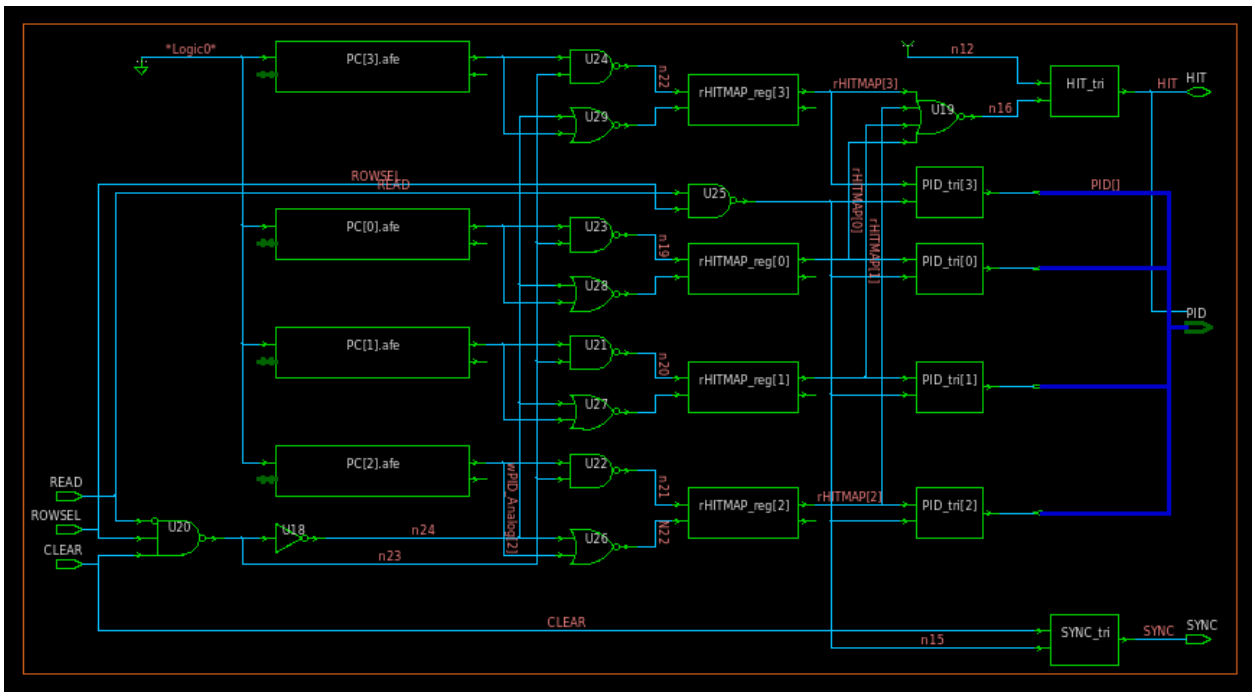


Figure 5.6: An image of the synthesised pixel region implementing only the readout-and-reset mechanism.

By looking at this figure, it is possible to see in details the working principles of the readout routine:

- `PC[n].afe` is the analogue front end; a black box from the digital design point of view

never considered from the timing, power, and area points of view;
- `rHITMAP_reg[n]` are the latches that register the current hitmap of the pixel region and save them waiting for the pixel readout and reset;
- the `NOR` and `NAND` at the center of the figure (from `U22` to `U29`) are used to both write on the latches and reset them when avoiding potential collisions
- `U20` is a `NAND` gate that checks when `READ` is low and both `ROWSEL` and `CLEAR` are high; it is used to trigger the internal reset of the latches;
- `U25` is the `NAND` gate that triggers the `PID` signal broadcasting when both `READ` and `ROWSEL` are high;
- `U19` is the `NOR` gate that produces a low hash value when at least one of the pixels in the pixel region is hit;
- `PID_tri[n]` are the tristates used to write on the shared `PID` buffer the hitmap value when the logic requests for it;
- `SYNC_tri` is the tristate used to write on the shared `SYNC` buffer at the correct time;
- `HIT_tri` is the tristate used to write on the shared hash lines (that are not represented here as must be added manually in the place-and-route phase, as shown in Section 5.1.4) when there is at least one hit in the pixel region.

5.1.2 Column State Machine (CLSM)

The readout routine must be performed by a syncrhonous module in the periphery, called column state machine (CLSM), responsible for both the correct broadcasting of the signals shown in Figure 5.4 (i.e., the `ROWSEL`, the `READ`, and the `CLEAR`), and the latching of the hitmap coming from the pixel (`PID`). There is one CLSM for each column in the array. In this implementation there are 16 columns in a sector, hence 16 different CLSMs in a sector for a total of 256 CLSMs in the whole array. Even though there are 16 finite state machines in a sector, the method used to select the correct pixel forces only one of them per sector to be active at a time, or the wrong pixels might answer to the request. When the hash decoder (described in Section 5.1.5) provides a new address to test, the CLSM of the corresponding column (the corresponding horizontal coordinate) answers and decides whether the address must be tested or not. It is the CLSM that tells the decoder that the readout-and-reset routine is over and a new address can be provided whenever available. This state machine has 7 different states, which implement the readout and check whether the read address was correct or a ghost. The automata scheme can be seen in Figure 5.7.

The state machine stays in `IDLE` until a signal arrives from the hash decoder asking for the readout of a pixel region in this column. When this happens, the machine can either start the readout procedure moving to the `READ` state or it can refuse the address by moving to `GHOST`. The latter happens when the proposed address was recently tested and revealed to be a ghost address. This procedure is necessary as it prevents some unpleasant loops found in simulations: going back and forth trying to read multiple ghosts[1]. If the address is not to be skipped, the next three states last one clock period only, and define the three

[1]This routine has been verified as safe also in the peculiar situation where the initial ghost actually has some new data, as `GHOST` is visited only once per address.

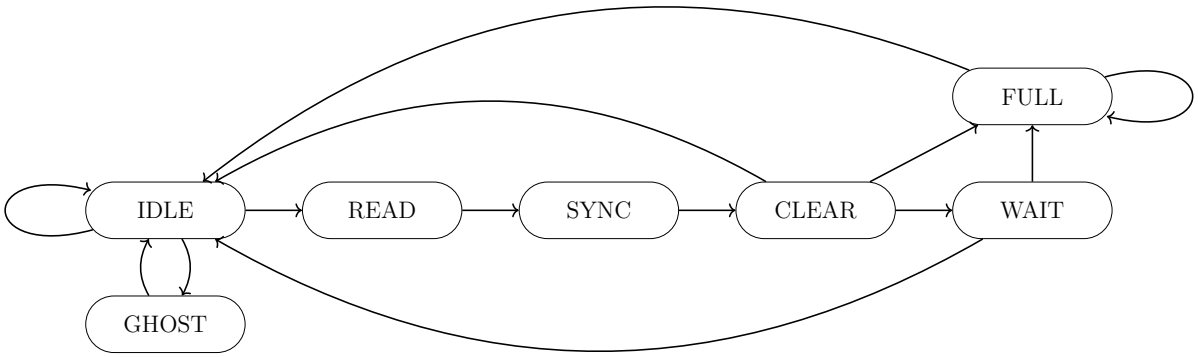


Figure 5.7: The automata of the column state machine. Only the transitions between states are shown.

timing regions necessary for the readout-and-reset: they are `READ`, `SYNC`, and `CLEAR`. The state after `CLEAR` is decided based on whether the data have reached this state machine or not. If the data reached the periphery, the following state can either be `FULL` if the readout address was correct or `IDLE` if the address was a ghost. `WAIT` state is used to wait for data arrival before choosing among these two states. This is particularly important when the array is relatively high or the periphery clock is fast, hence the time necessary for the data to reach the periphery varies from one to several clock cycles (actually, in all the possible architectures discussed in this thesis, this is necessary, at least as a safety net). If the machine is `FULL`, it waits for an external readout by another module, called DataGatherer (Section 5.1.7), in order to return to its `IDLE` state. Data arrive together with `SYNC` signal, which positive edge is used to enable storing the current value read on the `PID` bus.

Concerning the timestamp to associate to the hit just read, in this implementation there is a counter in the periphery that runs on a clock which period can be adjusted by setting a proper parameter. Whenever a valid address is produced from the decoder, the CLSM reads the current timestamp as provided by the counter and saves it internally. If the address was correct (i.e., there actually is a hit), the timestamp information is associated to the hit address. Otherwise, if the address was a ghost, then the timestamp is dropped, and the state machine waits for the next valid address. This means that the actual timestamp associated with the hit is the moment when the given address was produced by the decoder. On the one hand, the more bits are used for this timestamp, the higher is the power consumption; on the other hand, if too few bits are used it would be hard to reconstruct, after the serialization, the correct timestamp as a whole period might have passed. The number of bits to actually use depend on the implementation; for now a value of 8 bits is chosen to have a reference with ARCADIA-MD1 chip that suits most applications with a 80 MHz clock.

In Figure 5.8 an example of a readout cycle is shown. States are shown together with the signals that travel toward the pixel region to address. These three signals are driven by the CLSM itself, while the clock is not propagated to the array.

Other possibilities are available in terms of states, depending on whether the pixel prox-

imity to the periphery, whether DataGatherer is currently busy or the address is skipped (see Figure 5.9).

The signal provided to the decoder asking for a new address is named `NEXT`. In the `GHOST` state, it goes high as the periphery should focus on a different address as soon as possible; the same happens in both the `WAIT` and the `FULL` states. If the machine reaches one of these states, then the unwanted collisions between vertical and horizontal signals are no longer possible considering the array physical parameters. Every readout takes three clock cycles and there is an additional dead cycle; a different readout can be performed every four clock cycles (if addresses are available from the decoder).

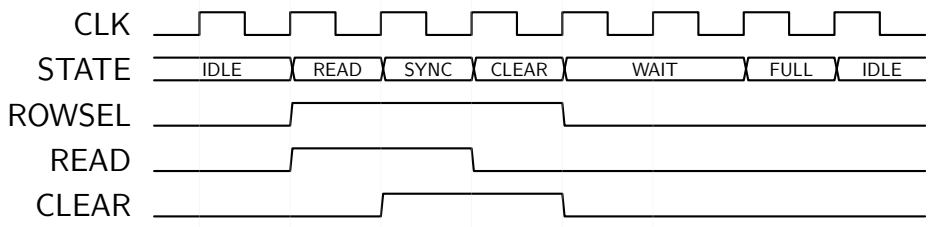
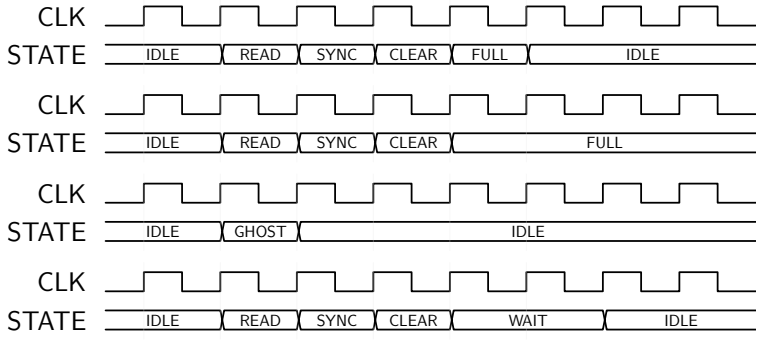


Figure 5.8: A standard readout cycle from the CLSM.



Readout when the pixel region is close to the periphery.

Readout when DataGatherer is busy and `FULL` state lasts longer.

Readout not performed because the address was recently tested.

Readout of a ghost address, CLSM goes back to `IDLE`.

Figure 5.9: Some examples of readout routines.

5.1.3 Column

The column does not actually implement any gate, but is made up of a set of pixel regions aligned along the y axis. All shared lines are implemented in the column and their connections from one pixel region to an adjacent one are implemented.

The interconnection among pixel regions and among columns is customised case by case, and the RTL code is generated via a C++ software that creates the Verilog code. In fact, while most signals are easy to propagate through the column, others are not as easy to implement. For example, the `READ` signal goes bottom-up and the `PID` signal goes top-down. These signals are easy to properly connect, as the connecting pattern is repeated throughout all the pixel regions, and do not require for customisation. The `ROWSELECTION` lines, on the

other hand, sometimes reach the pixel region from below, and other times from left or right. This opens up a plethora of possibilities. Once these rules are established, the RTL code is generated in a custom fashion for every pixel region considering its position in the array. The same holds for hashing the signals discussed in next section.

5.1.4 HASH code propagation

The most delicate aspect in the implementation of this architecture is the propagation of the hash code, therefore a lot of work has been devoted to the development of a scheme to move these signals around the array as necessary. To accomplish this task, it was decided to focus on occupying as little space as possible (thus enabling pixel shrinkage), on making the routing algorithm scalable (suitable for bigger sensors), and on creating a way to guide commercial place-and-route optimisation algorithms.

A rectangular hash array

A first issue is how to **reassemble** pixel regions in a rectangular device; i.e., passing from a 64×64 square array to a 16×256 rectangular one while maintaining the mathematical properties of the projections. The positions of the addresses can be reassembled as desired, but the projection relationship must be conserved: if in a square device there is a pixel with four specific values for the four projections, when the rectangular array is created there must still be somewhere a pixel with the exact same set of projected values, and this must be valid for every address in the array. This method is feasible only if the rectangular array considered can be mapped into a square one (the total number of pixels must be a perfect square).

For this project, two different ways to mathematically reassemble these addresses have been explored: one relies on the index, the other relies on the position. For editorial reasons, the figures report a 8×8 array reassembled in a 4×16 array, but all equations presented remain valid for any array size and any aspect ratio, including the one actually implemented. All the equations that are given in this section use the index of an address: an integer starting from 0 on the bottom left of the array and growing first towards right, then up. A representation of the index in a square and in a rectangular array can be seen in Figure 5.10. Mathematically, the index can be calculated as a function of the pixel region coordinates (used in the mathematical approach in Section 4.3) as:

$$i = x + Cy \tag{5.4}$$

with C as the number of columns in the device. This definition is valid for both square and rectangular arrays.

In the first reassemble possibility, called **index-driven**, the very same relationship between the index of a pixel and the hash values produced by such a pixel is kept. The result can be seen in Figure 5.11a. Considering that the side of the square array is S (in

56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

(a) Array indexing in the case of a square array.

60	61	62	63
56	57	58	59
52	53	54	55
48	49	50	51
44	45	46	47
40	41	42	43
36	37	38	39
32	33	34	35
28	29	30	31
24	25	26	27
20	21	22	23
16	17	18	19
12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

(b) Array indexing in the case of a rectangular array.

Figure 5.10: Array indexing. Equation 5.4 can be used both for square and for rectangular matrices.

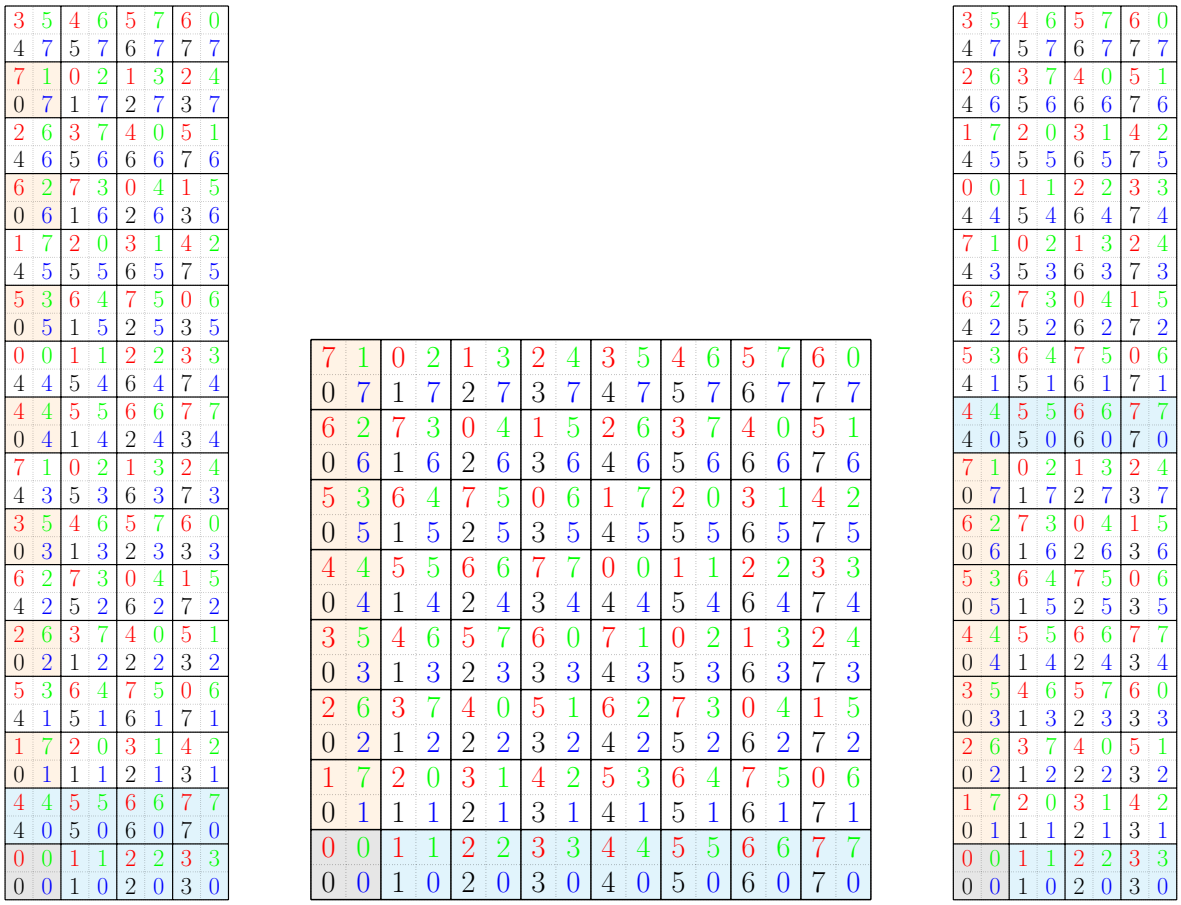
Figure 5.11a $S = 8$), the four projections can be calculated as:

$$\begin{aligned} P_X &= i \,\%\, S \\ P_Y &= i \,//\, S \\ P_U &= (P_X + P_Y) \,\%\, S \\ P_V &= (P_X - P_Y) \,\%\, S \end{aligned} \tag{5.5}$$

where $\%$ is the module operation and $//$ is the integer division operation. This definition works for both square and rectangular arrays with an arbitrary size and aspect ratio.

In index-driven reassemblement, the diagonal projections (P_U and P_V) connect pixels relatively far from each other, while in the square array these projections connect diagonals (in some cases broken). This aspect can be restored by proposing an alternative reassemble algorithm called position-driven.

In the **position-driven** reassemblement, P_X projections are kept close to one another (in index-driven they are distributed) while P_Y are separated (they are close in index-driven).



(a) Index-driven. **(b)** The original square matrix. **(c)** Position-driven.

Figure 5.11: The two ways to try and reassemble pixel addresses on a rectangular array. In black P_X, in blue P_Y, in red P_U, and in green P_V. Color shades used to show where the first row and column in the square case are mapped for rectangular arrays.

The projections can be seen in Figure 5.11c, where the following equations are used:

$$\begin{aligned} P_X &= i \,\%\, S + S(i \,//\, (SC)) \\ P_Y &= (i \,\%\, (SC)) \,//\, C \\ P_U &= (P_X + P_Y) \,\%\, S \\ P_V &= (P_X - P_Y) \,\%\, S \end{aligned} \tag{5.6}$$

where C represents the number of columns in the array, that is, the width of the array. Just like in the previous reassemblement, also in this case the mathematical properties of the projections are conserved, as the addresses, with their own projections, are just re-distributed in a rectangular shape.

In Figure 5.11, the color shades are used to guide the understanding of the reassemblement operation: the first row in the square is marked as cyan, while the first column is marked as orange. In the index-driven approach, orange is split and cyan is kept close; in the position-driven approach it is the contrary. **The position-driven approach was used**

for the implementation (Figure 5.11c and Equation 5.6) as it made the connections for P_U and P_V easier: pixels that share the same hash bits are closer and, as shown through this section, these two projections are the trickiest to implement physically.

Reference column

Each hash bit must reach the periphery in order to be used for the address guessing operation. For each hash bit there is a **reference column**, defined as the column where, at the bottom of the array, the hash bit enters the periphery. For the bits of the P_X projection, it is quite straightforward to understand the correct reference column, both in a square and in a rectangular array, as each bit runs in a column only. Concerning all the other bits, the reference column can be chosen arbitrarily. In this implementation, all the projections (P_Y, P_U, and P_V) follow the natural P_X. This means that, for example, by looking at an array like the one in Figure 5.11c, the first column (leftmost) is a reference column for the following hash bits:

$$\begin{aligned} P_X &= \{0, 4\} \\ P_Y &= \{0, 4\} \\ P_U &= \{0, 4\} \\ P_V &= \{0, 4\} \end{aligned} \tag{5.7}$$

The number of hash bits for which a column is a reference column depends only on the aspect ratio, as the number of hash bits per hash word is set to be the side of the reference square array S. In a square array, there are N hash bits per hash word to distribute in N columns; hence, one bit per projection per column works. In the drawings presented here, with an aspect ratio of 4:1, each column is a reference for two hash bits per projection. In the hash implementation described, where the sector is 16×256 super-pixels and the aspect ratio is 16:1, each column is a reference column for four hash bits for every projection. Although reference columns are used to properly distribute the hash values at the bottom of the array, this does not mean that every hash line runs only in its reference column (this is, in fact, true only for P_X).

Hash code inside a pixel region

As anticipated in Section 5.1.1, upstream of the hash network, there must be an `OR4` gate having as input data from the four pixels composing the pixel region. In this section, only the propagation along the network (after the hashing signal was created) is treated; for the sake of representation, such an upstream gate is not actually drawn here, but it is present nevertheless.

The pixel region layout was developed to be *smart*; that is, the layout (in terms of gates and metals placement) is identical for each pixel region in the array, making it easier to generate large area sensors, the only element changing from address to address is a fixed

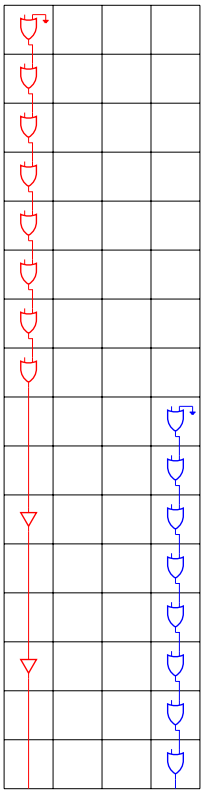
set of vias, which are programmatically enabled (via is actually present) or disabled (via is omitted) to determine the pixel region function in the array. The placement of each via is determined by a C++ algorithm using as only input the index of the pixel region. This software is fully customisable and can be easily adapted to different-shaped arrays.

X projection

P_X is introduced in previous chapter. In a square array, it would simply associate an address with its column number. Considering that that the implemented sector is shaped as a rectangle taller than wide (aspect ratio 16:1), to obtain the correct number of projected values P_X must project together only part of the actual column, as there are 64 possible different values for this projection (exactly one fourth of the 256-high column). In figure 5.12a an example of P_X projection of a smaller array (4×16) can be seen (as discussed before, position-driven reassemblement is pursued).

4	5	6	7
4	5	6	7
4	5	6	7
4	5	6	7
4	5	6	7
4	5	6	7
4	5	6	7
4	5	6	7
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3

(a) P_X projection in a rectangular array. In red and blue, hash bits which projection is shown on the right.



(b) Gates needed to implement some example hash bits of the P_X projection.

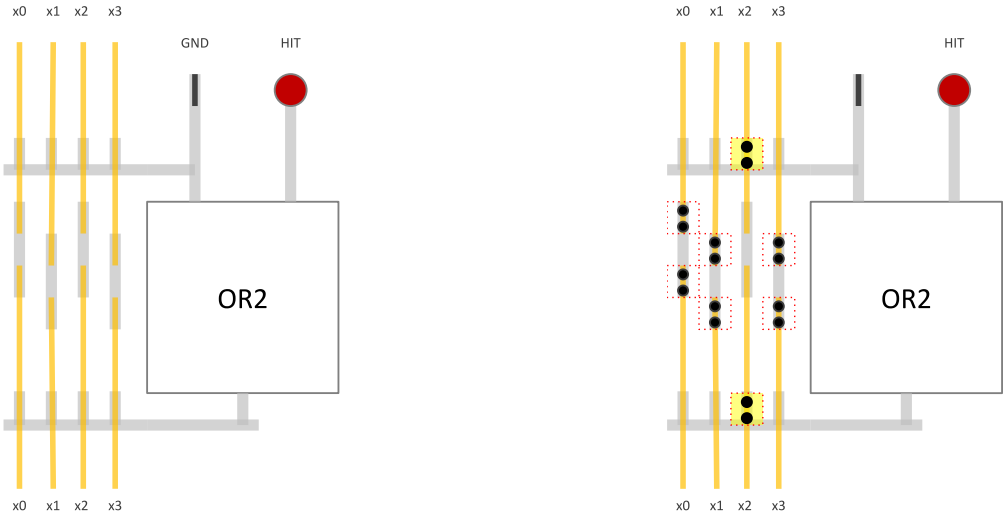
Figure 5.12: Example of implementation of P_X on a rectangular array. In this case, each column is divided in 2 parts connected to 2 different hash bits (as these figures have a different aspect ratio). In the taller sector actually implemented, there are 4 different parts per column. `OR2` are used together with `BUF`.

For this hash projection, the signal is always travelling vertically top-down, and regions

where the hash line is actually connected to the pixel region buffer are interspersed with regions of simple transmission of the signal towards the bottom of the array. This actually translates into two different gates used: `OR2` in the regions that share the correct projection and `BUF` (buffers) in the regions projected on different values, but where the hash signal needs to pass. In Figure 5.12b it is possible to see a representations of the gates used to implement this projection in two cases.

A buffer must be used every time the wire becomes too long, and it is convenient to break the capacitance load for timing optimisation (see Section 4.1.1) [107]; a repeated pattern is implemented inside the pixel region, and the automatised customisation algorithm connects it to properly to a given hash lane. The repeated pattern was created to buffer a signal every 16 pixel regions, hence the signal can be buffered every 800 µm. This value has been chosen after a study on the timing and power performances of the library gates (reported in Section 5.2.1), and the value is the same used in the case of ARCADIA-MD1, developed in the same technology.

Concerning the actual placement of the `OR` gates in silicon, the pixel region needs to catch one of the signals coming from the region above and `OR` it with its internal output before propagating it towards the periphery. To generalise this, the placement of the gate is as in Figure 5.13a. The yellow lines represent the four global lines going from top to bottom (in these lines the hash bits, that this column is a reference column of, run); grey lines are local to the pixel region and connected to the `OR` gate. Every pixel region intercepts one of these yellow-represented lines, uses it as input, and propagates it downward. Where yellow and grey superimpose, as well as where ground is represented, vias can be used to move the signal from one plane to another. In Figure 5.13b it is possible to see an example of the implementation of these vias.



(a) Generic repeated pattern for P_X projection.

(b) Some vias represented for a pixel connected to x_2 and propagating down all the other lanes.

Figure 5.13: Generic pattern to repeat and one implementation for P_X.

Every pixel region has to implement what is called a **move**. There are several possible

moves, and each move is translated into a set of vias that must be enabled in the proixmity of the OR gate. Each move is identified by a code; in the P_X case it is calculated as:

- η: a number identifying the position in the column where the hash bit of this pixel runs. It is implemented by one via among the four bottom ones. There is always one and only one of these vias;
- I: decide which lane drives the leftmost input of the gate. Can be set to G to be referred to ground;
- $ABCD$: for each value set to one, the corresponding signal coming from above is propagated toward the periphery by using the couple of vias next to the OR2 gate.

At the end, a move is fully described by these parameters. By using the code η_I_ABCD it is possible to understand where vias should go for this pixel region. For example, Figure 5.13b corresponds to 2_2_1101. An authomatised algorithm understands the placement of the necessary vias for every pixel region in the array; this algorithm is fully parameterised and may easily be adapted to work with different shapes and alternative layouts, given the replicated design (like the one in Figure 5.13a). The output of this algorithm (i.e., the move code) is then used as an input to constrain the place-and-route in the commercial software to guide the routing of the hash lines.

Y projection

Also P_Y projection is introduced in the previous chapter. In a square array, it would simply associate an address with its row number. Considering that here a rectangular sector is implemented, and it is higher than wide with an aspect ratio of 16:1, to obtain the correct number of projected values, P_Y must project on the very same bit different rows, as there must be 64 different values (exactly 4 rows in the implemented architecture, 2 in the smaller editor-friendly one).

In Figure 5.14a an example of the P_Y projection of a smaller array can be seen. Unlike the P_X projection, in the P_Y projection gate type and orientation[2] must be customised depending on the position of the pixel region. In fact, to propagate horizontally the signals, each hash bit for this projection must travel horizontally (left or right) to its reference column and then move downwards to reach the end of column. This process can be seen in Figure 5.14b where the gates used for the connection of one hash bit are drawn.

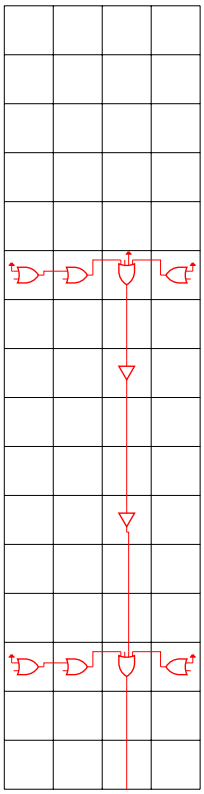
Also for this projection an algorithm has been developed to automatise connections between pixel regions. In this case, more moves are necessary, as the signal is sometimes propagated towards left, towards right, or downwards. Moreover, whereas an OR2 is enough when the signal is propagated horizontally, an OR4 is necessary in the reference column. To save in terms of complexity, OR4 are used in all cases, and in most regions some inputs are grounded. The repeated pattern in any pixel region is shown in Figure 5.15a.

This is the first projection requiring long horizontal connections: its usage makes the

[2]Figure 5.15 clarifies the meaning of **gate orientation**. The orientation depends on how the input and output pins are connected.

7	7	7	7
6	6	6	6
5	5	5	5
4	4	4	4
3	3	3	3
2	2	2	2
1	1	1	1
0	0	0	0
7	7	7	7
6	6	6	6
5	5	5	5
4	4	4	4
3	3	3	3
2	2	2	2
1	1	1	1
0	0	0	0

(a) P_Y projection in a rectangular array. In red, projected values propagated by the figure on the right.



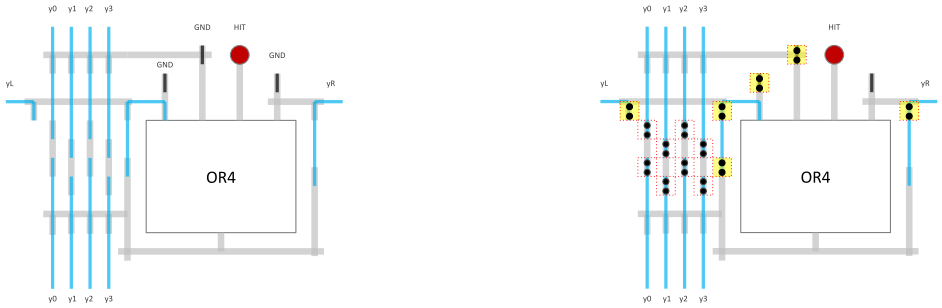
(b) Gates needed to implement an example hash bit of the P_Y projection.

Figure 5.14: Example of implementation of P_Y on a rectangular array. In this case, two different rows give the very same projected value. Different gates must be used and, in some cases signal goes left-to-right, while in other cases it goes the other way around.

actual place-and-route harder, as it is more complicated to fit horizontal signals crossing the whole section; nevertheless, it is necessary to implement this architecture. In the hash sector implementation discussed here, there are two horizontal signals crossing the whole array: the P_Y propagation and `ROWSEL` signal used for readout-and-reset.

Depending on the vias position, the input bits used and the output direction vary. In Figure 5.15b an example of an implementation is shown. The code identifying the move is:

- η to drive the gate output. Can be set to a number between 0 and 4, or L or R to send the bit left or right. In the case of a number, a via must be enabled in the bottom left, if L three vias must be enabled to connect the output to what is called yL, if R two vias must be used to connect the output to what is referred to as yR;
- M to choose the signal to use in the leftmost input pin of the gate. It can have two possible values: G for ground or L for left. In the former case, one via is enough; in the latter, three are needed to connect the input to yL. It is impossible to have both $\eta = L$ and $M = L$, as the left signal must be either an input or an output;
- N to choose the signal to use on the second left input pin of the gate. It can either be



(a) Generic repeated pattern for P_Y projection.

(b) Some vias represented for a pixel propagating its signal from right to left.

Figure 5.15: Generic pattern to repeat and one implementation for P_Y.

G for a connection to ground (and one via is enough) or a number to catch the signal from above (and yet one via is enough);

- O to choose the signal to use in the rightmost input pin of the gate. Can have two possible values: G for ground or R for right. In both cases, one via is enough, connecting either the input either to ground or to yR. It is impossible to have both $\eta = R$ and $O = R$;
- $ABCD$: for each value set to one, the corresponding signal from above is propagated toward the periphery by enabling the pair of vias next to the gate, just like in P_X.

The code is then given as η`_MNO_ABCD`. In the case of Figure 5.15b the code would be `L_GGR_1111` as this implementation transports the hash bit from right to left while propagating all vertical signals.

Diagonal projections

The last projections chosen for the implemented architecture are the diagonal ones, that are P_U and P_V, defined in a square array as:

$$P_U = (1, 1) \qquad P_V = (1, -1)$$

In a rectangular array, the approach highlighted in Figure 5.11c was chosen to ease the routing to implement these projections; the two projections are shown, in the editorial-friendly smaller array, in Figure 5.16.

Typically, CMOS PDKs (**P**rocess **D**esign **K**its) do not allow for 45 degree routing, but only vertical and horizontal ones are allowed. Therefore, to connect these hash lines properly, a way to move the signal diagnoally had to be implemented. This approach was naively called *corners*-approach. The idea is to keep the connection on one of the sides of the pixel region and use a third pixel to change the direction of the signal. A representation of this can be seen in Figure 5.17. The third region does not perform any active action on the incoming signal, it is used only for its spatial position. When the gate schematics is presented (like in Figure 5.18), there is yet one signal on the left side and one on the right side, meant to travel diagonally.

3	4	5	6
2	3	4	5
1	2	3	4
0	1	2	3
7	0	1	2
6	7	0	1
5	6	7	0
4	5	6	7
7	0	1	2
6	7	0	1
5	6	7	0
4	5	6	7
3	4	5	6
2	3	4	5
1	2	3	4
0	1	2	3

(a) P_U projection in a rectangular array.

5	6	7	0
6	7	0	1
7	0	1	2
0	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7
1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7
5	6	7	0
6	7	0	1
7	0	1	2
0	1	2	3

(b) P_V projection in a rectangular array.

Figure 5.16: Both P_U and P_V projections are diagonal along the array. In these figure, their representation in a 4×16 array. Their shape is peculiar, as many diagonals (all but bit 3 and bit 7 in P_U case and 0 and 4 in P_V case) *break* at some point, not crossing the whole array. Colors on the left hand side are added as a guidance for Figure 5.18.

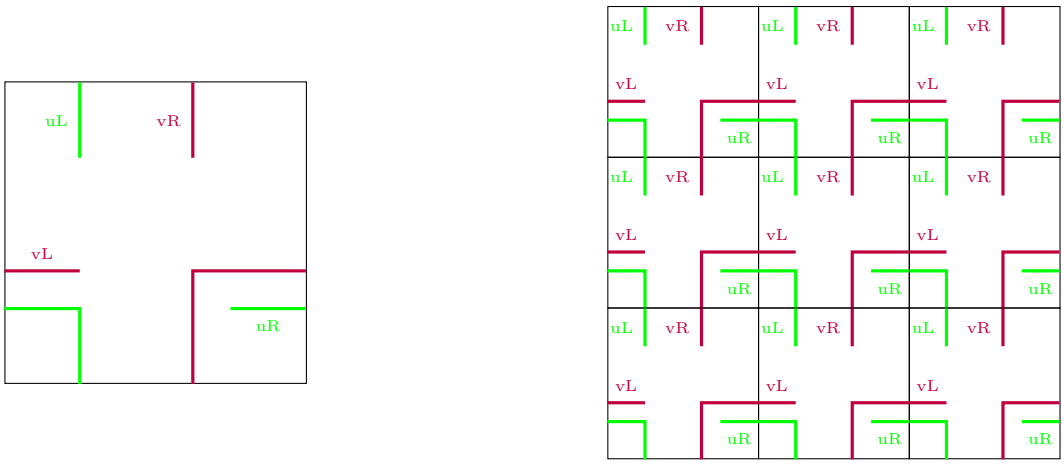
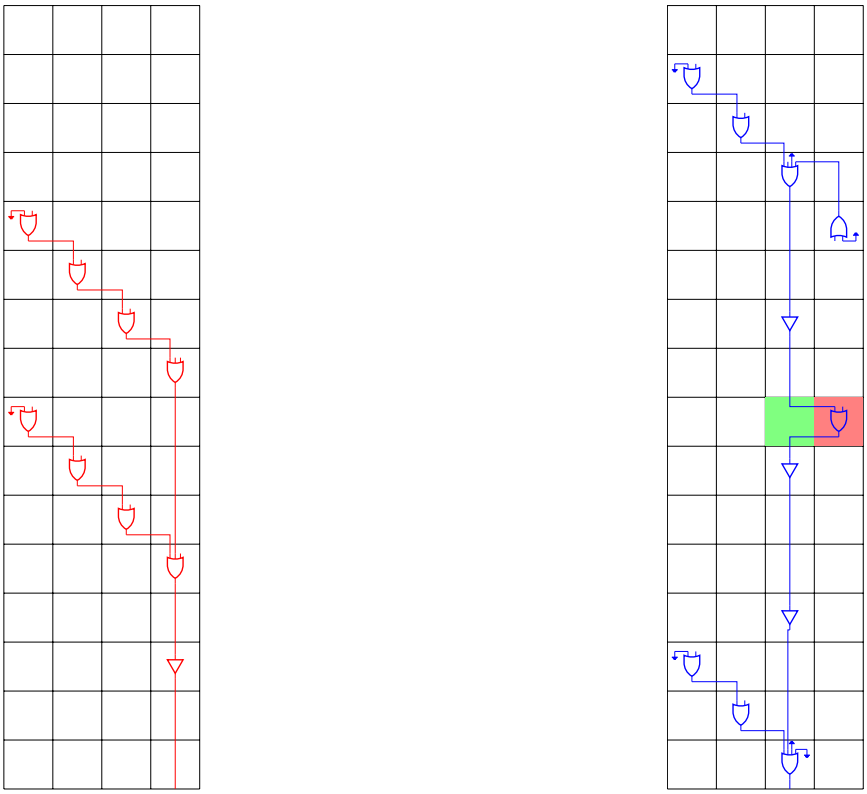


Figure 5.17: A representation (not to scale) of a pixel region with the lines needed for diagonal signal passage, called corners. On the left, the needed routing on a pixel region; on the right, a replica of the scheme that allows diagonal signal movement. Both P_U and P_V, which move on different diagonals, are shown.

This section starts discussing the P_U projection; at the end a description what must be done for P_V is given: the two projections are similar and the approach to implement them both is identical, the routing is only slightly different. Starting from projected bit 7 (or 3, which has a similar shape) on the small array, according to the performed choice of the reference columns, this projection should run on the very last column, and connected regions are distributed along two complete diagonals side-to-side. `OR` gates and `BUF` are enough, and in all cases the signal travels from left to right and from top to bottom. These hash bits were originally running as the diagonal of the square array (see Figure 5.11b), or the diagonal was cut exactly in half. The idea of the placing of the gates can be seen in Figure 5.18a.



(a) P_U gate implementation for bit 7.

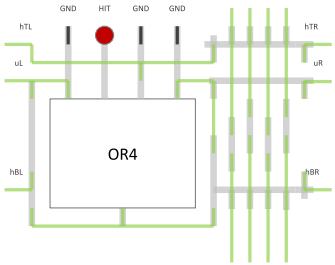
(b) P_U gate implementation for bit 2.

Figure 5.18: In diagonal projections (here P_U) different approaches must be used to connect correctly the hash network. A diagonal move is necessary. In some cases, diagonal is always from left to right (Figure 5.18a), while in other cases there is also the need of a right to left move (Figure 5.18b). Other more complicated moves are needed and described on text. In color, hooking and hooked, described in text.

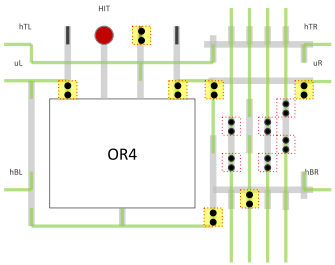
Implementing projections different from 3 and 7 as shown in Figure 5.16a is more complicated; there is a full diagonal and a broken diagonal (see, for example, projection 2, highlighted in blue). In this case, `OR` gates and `BUF` are still enough, but there is a position in the array where the output of the usual `OR` gate must travel horizontally. This can be seen in Figure 5.18b. This peculiar move is called **hook**. During this move, the signal is sent from its reference column to the adjacent one, caught, used as an input of the `OR` gate, and sent back to the reference column. This is explained in detail later.

Concerning the differences between the small array drawn and the larger array actually implemented, the only difference is the number of times hook operation is necessary. Actually, whereas in a square array only one hash bit would not require for hook (all other diagonals are cut in two parts), in a 4:1 there are 2 bits where hook is not necessary, and in a 16:1 there are 4. The number of hooking rows (that is, the rows where hooking operation is performed) grows with the exact same factor: in a square array it is 1, in a 4:1 they are 2 and in a 16:1 they are 4.

Looking at Figure 5.18a, the need for local horizontal connections can be spot (while P_X did not have horizontal connections at all and P_Y had array-long horizontal connections). The list of moves for these projections is the largest of all the ones implemented in this project. As in the P_Y implementation, although smaller gates could sometimes be used, only `OR4` have been used in every pixel to ease the implementation. P_U connects to the same bit top-left and bottom-right regions, the inputs may come diagonally and vertically, and sometimes hooking is necessary. The generic gate pattern can be seen in Figure 5.19a, whereas a concrete move implementation is shown in Figure 5.19b. While the two lines uL and uR are used for diagonal moving of the signal, the two signals labelled with the letter h are used for hooking. hTL stands for *Hook To Left*, hBL stands for *Hook Back from Left* and the same holds for the corresponding signals on the right side.



(a) Generic repeated pattern for P_U projection.



(b) Some vias represented for signal collection on top border.

Figure 5.19: Generic pattern to repeat and one implementation for P_U.

The code used to describe the move performed by each pixel region (hook is described later) is:

- η to drive the output of the gate. Can be set to a number between 0 and 3, L or R to drive left or right, H when hooking operation is necessary. When η is a number, enabling two vias below the gate is sufficient to send the signal on the correct line. If η is L, two vias must be enabled: one on the bottom left and another close to the uL line. If it is R, then three vias are necessary: two just to the right of the gate and one close to uR. With H, two vias are enabled on the bottom right to link the output to the hBR line;
- M to choose the signal to use in the leftmost input pin of the gate. Can be either G for ground (one via is enough) or L for left (and yet one via is enough);
- N to choose the signal to use in the second rightmost input pin of the gate. It can be

G for ground (and one via is enough), H for the hooking signal from the right, or a number to catch the signal from above (and two vias must be enabled);
- O to choose the signal to use in the rightmost input pin of the gate. Can have two possible values: G for ground or R for right. This signal is just like the one for P_Y and the connection is implemented exactly enabling the same set of vias;
- $ABCD$ to choose whether to continue propagating the signals vertically, as already explained in P_X and P_Y case;
- h is used to send the hooking signal toward the adjacent pixel region. If equal to a number, it implies that this region is hooking, and there two vias are enabled on the top right of the scheme (one on the corresponding line, one close to hTR) to connect one of the vertical lines with the hooking line and two more vias on the same line on the bottom right to redirect the signal to the correct line. When this is set to G, it means that no hooking operation is performed, and no vias are enabled.

As usual, the code can be written as η`_MNO_ABCD_h`. In the case of Figure 5.19b the corresponding code would be `1_LGR_1011_G`.

Studying the hooking operation in detail, the two pixels regions involved get the names of **hooking** and **hooked**. A region is hooked when it must be connected to a hash bit but has currently no direct access to the reference column via other regions with the same hash bits; the hooking one is the adjacent one where the hash bit is currently running. In Figure 5.18b a hooking is represented in green and a hooked in red. P_U hooks always from the left side (hooking is always left with respect to hooked).

Another interesting fact is that hooks are necessary only on some rows of the array, and for every region of the row: by studying the projected values in Figure 5.16a it possible to see that all broken diagonals break in the middle of the array (lines 7-8). This property holds for bigger matrices and using different aspect ratios: there always is only a small set of rows that need to perform hooks, and there is always at most only one broken diagonal for every hash bit. Hence, the leftmost pixel regions of these rows are hooked, the rightmost ones are hooking, and all the pixels within the rows are both hooking and hooked for different projections. A pair hooking-hooked can be seen in Figure 5.20. Using all these moves, it is possible to physically implement the `OR` chain necessary to support the projection P_U.

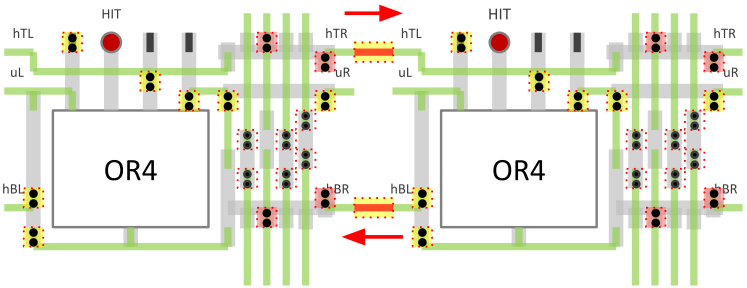


Figure 5.20: An example of a hooking-hooked (respectively on the left and on the right hand side) pair for P_U. The right hand side pixel region signal contributes to the bit currently running on the second line of the region to its left.

Concerning P_V, although the shape is the same, the repeatable pattern is horizontally

mirrored: the lines must travel on the left side of the pixel instead of the right side. This, in fact, makes it possible to perform the same moves that have been described for P_U: the diagonal movement (which for this P_V connects top-right and bottom-left), the collection that pushes the signal toward the periphery, and the hook. The code to identify these remains the same: η_MNO_ABCD_h with the same meaning. About the hooking operations, the same rows where P_U needs hooks must implement the hook for P_V as well.

5.1.5 Hash decoder (HDCD)

The hash decoder (HDCD) is a module in the periphery that guesses an address by looking at the hash code. There may be different approaches for guessing; in this very first hash architecture implementation the choice fell on a simple and safe one without the need of too much space. This module is clocked.

The hash decoding operation is performed based on the fact that two of the implemented hash projections directly provide the X and Y coordinates of a plausible address. For each clock cycle, a tentative address is produced by looking at P_X and P_Y using some proper priority encoders. Some combinatorial logic is used to calculate the hash bits that such an address would trigger under P_U and P_V. If both P_U and P_V hash words are compatible with a hit in the tentative address, then this address is promoted to valid, and is communicated to the corresponding CLSM for the readout. If this is not the case, then a mask will be put on the corresponding X hash bit, and in the next clock cycle the decoder repeats this operation with the next tentative address. Every time all currently high bits in the P_X hash word are masked, all these masks are removed and a new mask is placed on the tentative P_Y hash bit. P_Y mask is fully reset every time there are no more tentative addresses to study. This approach ensures that every possible address is visited; therefore, the logic can always find a valid address for every hash code. The two priority encoders (one for X and one for Y coordinate) provide the address of the currently lowest bit in the two hash words taking into account the masks; hence the tentative address is the leftmost among the lowest addresses possible[3].

In order to understand the time taken by such an algorithm to produce a valid address, it is necessary to study how many hash bits are actually high when the array population is fixed. This problem is a variation of the **birthday problem**, and its solution can be found in literature [130]. Considering one projetion only, the average number of high bits when there are m hits in an N-sized square array is:

$$\bar{H_b} = N - N\left(\frac{N-1}{N}\right)^m \tag{5.8}$$

This equation can be used to understand, as a reference, how many tentative pixels can be found in the lowest row before a valid address. There are, on average, $\bar{H_b}$ high bits in P_X, and likely only one of them is correct (there is a second-order correction given by quadratic

[3]Even this operation needs a correct tuning for the rectangular array, that has been performed.

terms). This means that on average the number of wrong guesses is equal to half the number of tentative addresses; i.e., the number of clock cycles needed to reach a valid address while exploring the space of tentative addresses is:

$$\bar{t} = \bar{H}_b/2 = \frac{1}{2}\left[N - N\left(\frac{N-1}{N}\right)^m\right] \tag{5.9}$$

For example, if there are 4 hits in the array 1.96 clock cycles are necessary on average; if there are 8 hits in the array 3.78 clock cycles are necessary and with 12 hits 5.51 clock cycles are necessary. Considering the application regime of the hash architecture (in particular the sparsification hypothesis), in good approximation, on average $m/2$ clock cycles are necessary to produce a valid address.

Going back to the physical implementation of this module, whenever the HDCD finds a valid address, it is communicated to the corresponding CLSM that saves the address. In the following clock cycle, HDCD starts searching internally for the subsequent address. This second address research continues while the first address is being looked at by the CLSM and the array cannot perform more readouts to avoid collisions. Then, after the CLSM broadcasts the `NEXT` signal, the second address is given by the HDCD, if available; otherwise, there will be no address to read for some time. This behaviour justifies a peculiar effect found in simulations, where the bottleneck for the readout speed is due to the time necessary to generate a valid address (discussion in details in Section 5.3.2). For this reason, alternative hash decoders are discussed in Section 5.4.

5.1.6 Configuration Manager (CFGSM)

Configuration management is performed by finite state machines in the periphery. Whereas for readout state machine (CLSM, Section 5.1.2) there is a state machine for every column, for the configuration there is only one state machine for every hash sector, that is, 16 in the whole array discussed in this section. This state machine is called CFGSM. When a configuration request reaches the chip, it is transferred to one of the 16 FIFOs responsible for the saving of configuration words for a sector. Then the state machine manages the request by executing the configuration commands one at a time. CFGSM performs three different tasks: it sends testpulses to the correct columns inside the sector, sets the global configuration bits, or change one of the local configuration bits in a set of pixel regions.

A configuration setup request

Configuration setup has a set of rules to function correctly and, to work as planned, the FPGA that drives the sensor must provide the correct word. A representation of such a word can be seen in Figure 5.21, while the bits are reported in Table 5.1.

Although these words can contain information on both testpulses, global configuration, and local configuration, only one of these options is performed by the configuration state machine. The priority list is first testpulses, then global configuration registers, and finally

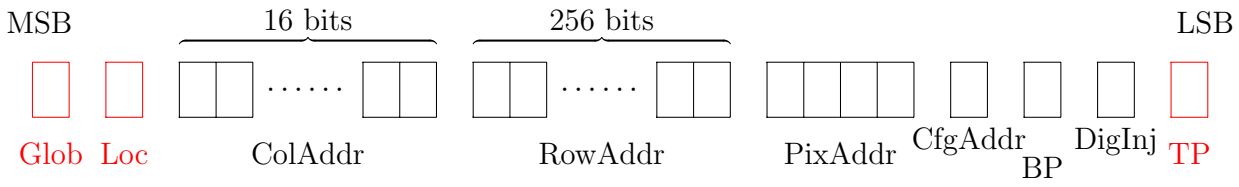


Figure 5.21: The representation of the 282 bits word used for configuration setup. In red, the bits that identify the type of this configuration word.

local configuration registers. This means that if the `testpulse` bit is one, a testpulse is sent regardless of the bits values in the `validGlobal` and `validLocal` positions of the configuration word.

Word name	Width	meaning
`validGlobal`	1	if high, this word is used to setup a global configuration register
`validLocal`	1	if high, this word is used to setup a local configuration register
`col`	16	the columns to configure (one-hot encoded)
`row`	256	the rows to configure (one-hot encoded)
`pixAddress`	4	the address of the pixel to configure inside the pixel region (one-hot encoded)
`cfgAddress`	1	the address of the pixel configuration bit register to set locally
`bypass`	1	the new value to set globally for bypass configuration bit
`digInj`	1	the new value to set globally for Digital injection configuration bit
`testPulse`	1	if high, this word is not actually used to configure part of the array but to send a testpulse

Table 5.1: Configuration words, starting from the most significant bit.

Possible operations are the following:

- Send a testpulse to some columns. In this case, the `testPulse` bit must be high, and the `col` word must have at least one high value. The testpulse is sent to all the columns in this latter word. All other bits are neglected (testpulses is always propagated to whole columns, not to the single pixel regions);
- Set the values of all the global configuration bits. This operation is performed if the `validGlobal` bit is high. The bits to set are read in `bypass` and `digitalInj` positions. There are actually two more bits propagated globally, and these are the ones used when the bypass signal is high in the pixel region. Also their setup is performed in this routine, and bits are read from the two LSB of `pixAddress`. If the configuration word is a global configuration word, then all the 4 global configuration bits must be set accordingly. It is possible to setup several columns in a sector at the same time by using the one-hot encoded column representation;
- Set Local configuration values to a given value. This is performed if `valodLocal` is high. The two bits that can be set via this command are the masking and the injection enable, and the values of these bits might not be used in the presence of a high bypass

global signal. In the case of local configuration setup, contrarily to what happens to global registers, only one of the bits is set by the configuration word. The address of the bit is read from `cfgAddress` (1 for masking, 0 for injection enabling), and the map of the bit to set inside the region is read from `pixAddress`. Several pixel regions can be configured at the same time, taking advantage of the fact that both the row and the column addresses are represented in one-hot encoding.

Configuration routine

Configuration words are digested by a state machine that reads them and performs the necessary actions. The state machine automata can be seen in Figure 5.22. It is a relatively easy state machine that is used to manage the waiting time to configure everything properly.

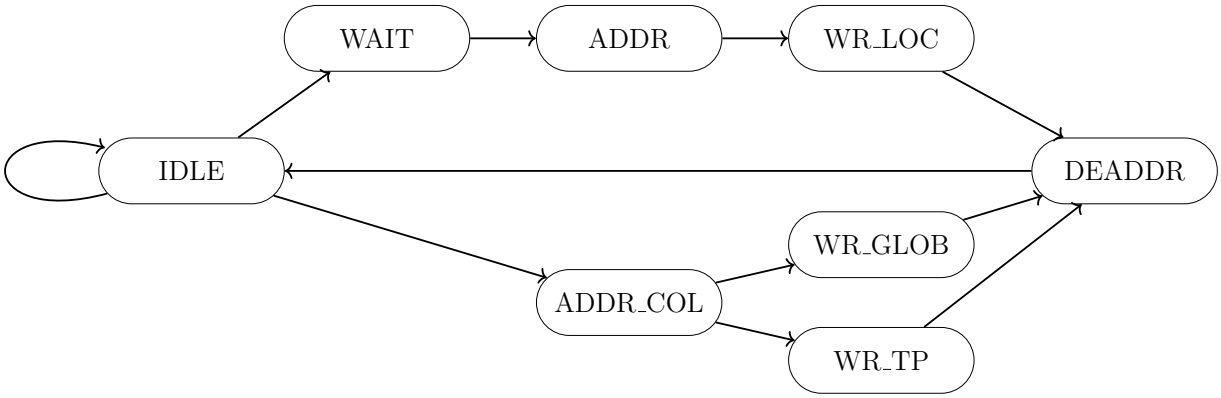


Figure 5.22: The automata of the configuration state machine. `WAIT` and `DEADDR` states are necessary to avoid collisions with readout routine. Global configuration and TPS need less dead time as there is no need to address a single pixel region in the array.

In particular, it is important for this state machine to not collide with the readout state machines. In this implementation, configuration has priority over readout; nevertheless, if there is an ongoing readout, the configuration waits for it to finish. Exactly as discussed for the readout, collisions are tricky also in the case of configuration operations: Only the task requested by one configuration word can be exhausted at a time on a sector, and no readouts can happen at the same time on the same sector. This is due to the fact that some lanes used for readout are also used to configure (such as the `ROWSEL` signal), thus reducing the overall number of lanes in the array.

5.1.7 DataGatherer

The DataGatherer module is a module used when there is data scattered in several different objects but a common interface is necessary. It was first written by the INFN Torino group and is currently implemented also in ARCADIA-MD1. In this physical implementation of the hash architecture, DataGatherer module is used to read data currently inside the CLSM

and reset them, putting them in a memory connected to the serializer. There is a memory for every sector, as there is a serializer for each.

The development of this architecture did not focus on data serialization, willing to use those that were already developed and thoroughly tested for ARCADIA-MD1 sensor in the same techonlogy.

5.2 Timing and power Performances

The hash architecture has been studied using an analytical approach and verified through Monte Carlo simulations. For the latter, timing performances had to be put as parameters of the simulations for it to work properly. Signal propagation speeds have been calculated by referring to the LFoundry imaging 110 nm technology [30] as a reference, with a *realistic* architecture design. Additionally, some possible timing issues in the algorithms implemented have been studied, and a simple calculation on the clock speed limit is presented. Furthermore, some early calculations of power consumption post-synthesis have been carried out.

The methods used to perform these calculations are described in Section 4.1.2. Three different array geometries have been analysed:

- **array** $\mathcal{A}$: 512×512 square shaped pixel with a pitch of 50 µm;
- **array** $\mathcal{B}$: with an aspect ratio of 1:16; a 128×2048 square shaped pixel with a pitch of 50 µm;
- **array** $\mathcal{C}$: with an aspect ratio of 1:16 and small pixels; a 128×2048 square shaped pixel with a pitch of 10 µm;[4]
- **array** $\mathcal{R}$: the array actually developed in SystemVerilog and described in Section 5.1 (the one actually implemented). The repeated module (a sector) has an aspect ratio of 1:16 and is composed of a 16×256 square shaped pixel regions with a pitch of 50 µm (this is the sector composed of the addressable pixel regions; then each pixel region is made of 4 pixels). A sector can be thought of as a standalone detector, as each sector is independent.

A different approach was used for what are called in this work **data lanes** and what are called **hash lanes**. A hash lane is a lane that transports the hash bits; hence, a lane intersparsed with `OR` and `BUF`. The number of hash lanes corresponds to the length of the hash code produced in the periphery. All other lanes are data lanes; therefore, according to the implementation discussed in Section 5.1, data lanes used for the readout are `READ`, `SYNC`, `CLEAR`, `ROWSEL`, and `PID` bus.

[4]The feasibility of such small pixels, although would need proper demonstration, may be possible thanks to the size of the digital architecture implemented, discussed in Section 5.1.1.

Long lanes segmentation

As highlighted in Section 4.1 in CMOS technology the delay on a lane can be reduced by using some buffers to reduce the overall product between the wire resistance and its capacitance. A study was performed to estimate the power consumption and delay on a data lane varying the distance between the buffers or the buffer strength.

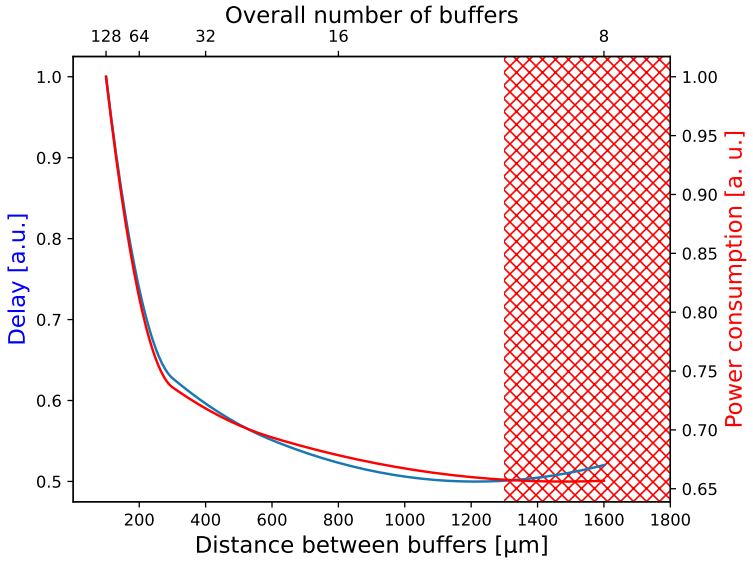


Figure 5.23: Power consumption and total lane delay as a function of the distance between the buffers. Forbidden region is due to the wires so long that the maximum driving capacitance is reached.

From Figure 5.23, calculated with a chosen buffer size and showing values in arbitrary units for non-disclosing reasons, it is possible to see that in the allowed region (where the total capacitance is low enough for this buffer strength) both the power and the delay improve by reducing the overall number of buffers.

In Figure 5.24, on the other hand, a comparison between buffers with different strengths is performed. Here a fixed distance of 800 µm is taken as a reference.

For the implemented design, a conservative value of 800 µm and a medium-sized buffer has been used, borrowing the implementation developed for ARCADIA-MD1 in the very same technology, that was shown as working correctly.

5.2.1 Timing performances

Data lane speed

In the simulations, only the vertical delay was considered, and the time needed to cross horizontally the array was neglected[5]. To study the time that a signal takes to reach a given

[5]In all this section but in the timing stability, that depends also on the horizontal delay.

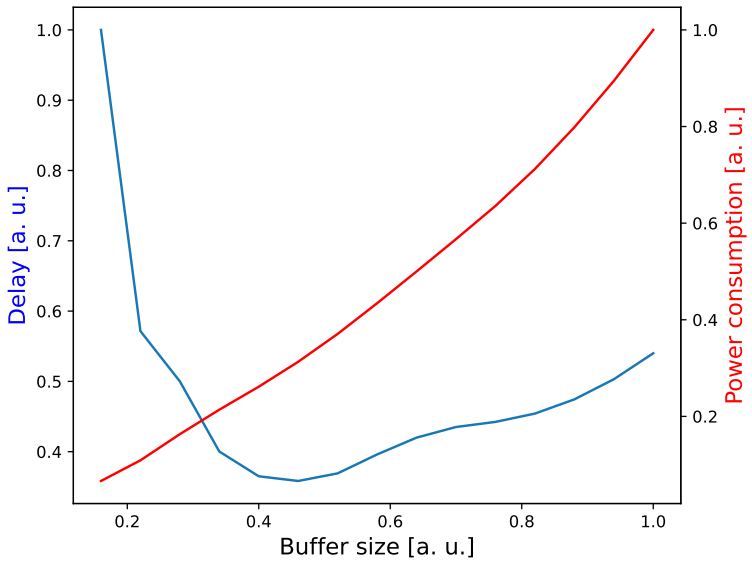


Figure 5.24: Power consumption and buffer delay as a function of the buffer strength (thus, its size). Here, a fixed distance between the buffers of 800 µm has been chosen.

position within the array, the approach used here is the same that was used in [120] and described in Section 4.1.2. The capacitance and impedance of every segment of the lane is calculated according to a pessimistically crowded place-and-route, taking into account the gates input capacitance at the expected operating point. The reference metal layers used for calculations where these where such long lines would realistically fit, and a plausible wire width according to previous designs on the very same technology (ARCADIA-MD1). Subsequently, the segment capacitance wass used to calculate the delay due to the buffer itself. Finally, the total delay was calculated by adding the delay due to all the wire segments present and the delay due to all the buffers used. Calculations are not reported here as the parameters used are covered by the already mentioned non-disclosure agreement. This delay was used to calculate the average speed (in a very conservative regime) and the result obtained is as follows:

$$v_s = 7\,\mathrm{ns\,cm^{-1}} \tag{5.10}$$

This speed was taken as a reference even though the idea of a constant signal speed is a mere simplification. Considering that the number of buffers is set given the array height and does not depend on the number of rows, this speed is considered the same for all the geometries discussed.

Hash speed

For what concerns hash lines, in this case the signal starts from a buffer in pixel and travels a long chain of `OR` gates to reach the end of column. In a square array, there is one of these `OR` gates for each row of the array, while in a rectangular array, there is an overall number of `OR` gates equal to the side of the hash geometry, and some space is covered by normal lines, interspersed with buffers and nothing more (see, for example, Figure 5.12a). Both

the number of gates and the actual line length (hence the delay) strongly depend on the position of the pixel region. Once again, a reference average speed was calculated in the various geometries taking into account only the pixel height (and not the actual number of OR2 between the pixel and the periphery). Also in this case a realistic use of a buffer every 800 µm was used where OR gates are not present. The calculated speed values v_H are reported in Table 5.2 at the end of this section and these signals are clearly slower than data ones.

Regarding timing stability for the hash code, hash lines are synced at the periphery. If anyhow there was a problem in the syncrhonisation of the different hash lanes, the worst that can happen is that the decoder might cycle through all possible masks. This would lead to a time and power loss, but the state is completely recoverable with no issues in a few cycles (this was simulated).

Overall timing stability and clock period

Hash architecture synchronous logic is relatively simple. The only three modules that require a clock signal are the column state machine (see Section 5.1.2), the address decoder (see Section 5.1.5), and the configuration state machine (see Section 5.1.6). All these modules sample their inputs on the clock transition; hence, time closure is relatively straightforward. Pixel regions, on the other hand, trigger their actions based on some combinatorial logic. This conservative approach strongly simplifies the timing constraints on the architecture, which does not rely on negative edges, nor it does request for a certain number of things to happen in the same clock cycle. The most delicate corner for timing is the expected concurrency of high ROWSEL and high READ (or CLEAR) signals, which trigger the readout-and-reset routine (described in Section 5.1). A non-ideal timing performance can lead to two unwanted effects: the correct region does not respond to the request, or the wrong region responds because of collisions between separate readouts.

If the region has to respond correctly, the three time windows discussed in Section 5.1.1 must be long enough for the combinatorial logic to work. This means that vertical signals and the horizontal signal cannot be delayed too much one with respect to the other. The vertical signals run straight from the CLSM to the region to read (and both take, at first approximation, the same time to do so), while the ROWSEL information moves first inside the periphery to the column where the row signal runs, then reaches the correct height and finally moves horizontally once again. By assuming that these signals have a similar signal propagation speed (thus, similar wire width and parasitic capacitance) v_S, with p the pitch of the array, the time between the CLSM transmission of a vertical signal and its arrival at the pixel region in the row r is:

$$t_V = (p \cdot r)v_s$$

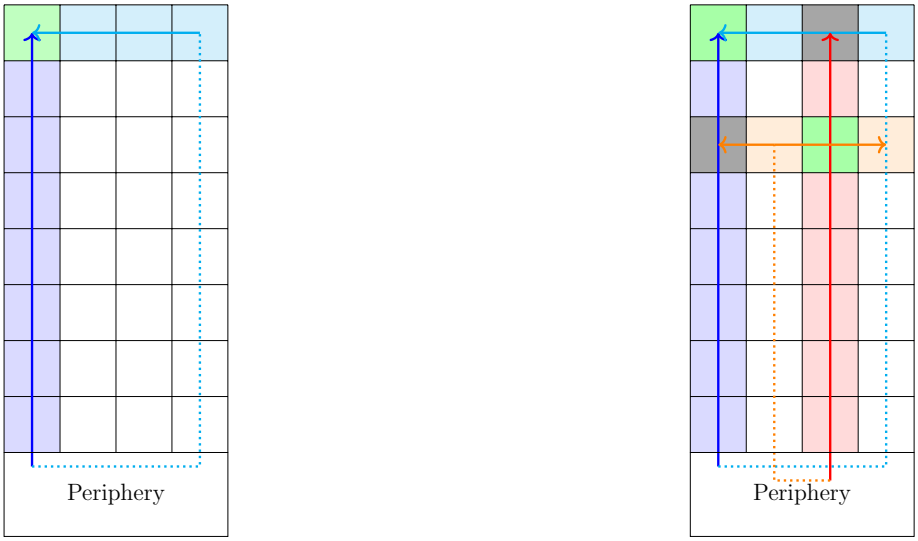
On the other hand, the ROWSEL signal, in the worst corner, crosses the array horizontally twice (once in the periphery and once in the correct row) as shown in Figure 5.25a. Therefore, the time delay in this case is:

$$t_H = (p \cdot r + 2W)v_s$$

with W width of the sensor. This means that the time difference in the worst case is

$$\Delta t_{VH} = 2Wv_s$$

and this Δt_{VH} must be smaller than a full clock cycle for the pixel to respond (see readout timing diagram, Figure 5.5; if the delay was greater than 1 clock cycle, the SYNC signal would be broadcast too soon, before the hitmap is actually stable, thus leading to a possible mistake in the address saving and an inefficiency seen as a missing hit and a fake hit).



(a) A readout mechanism picking a region in the array. The lane paths in the worse corner are depicted.

(b) Two different readouts on the same array (cold-coloured and hot-coloured).

Figure 5.25: The mechanism used to select a region for readout-and-reset, picking whole columns or rows. Lightly shaded, the regions selected by either a vertical signal (blue and red) or ROWSEL horizontal signal (cyan and orange). In green the region identified by the presence of two signals. In grey, the region that might be selected by mistake in an unsafe routine.

For the wrong region to respond, on the other hand, there must be a superimposition between the ROWSEL of a first readout and a vertical signal of a second readout. In the current implementation of the CLSM (Section 5.1.2) there is one dead cycle after the readout, that is, three clock cycles are used for the readout, and four clock cycles must pass between a readout and the subsequent one. with these assumptions, where the signal speed is constant for all signals used, the worst delay is still Δt_{VH}. Therefore, for the readout-and-reset routine to work properly, the clock period T must be:

$$T > 2Wv_s \tag{5.11}$$

Given the signal speed of $7\,\mathrm{ns\,cm^{-1}}$ (Equation 5.10), the maximum clock period can be calculated and is reported in Table 5.2. This table must be taken with grain of salt: if a clock of 2 ns in array $\mathcal{R}$ was implemented, there will be no problems related to the issue just

discussed; nevertheless, many of the assumptions made to perform such calculations would fail with such a fast clock (for example, in this calculations the the parasitic capacitances were considered constant through the array, and the signal movement speed was considered fixed both in the vertical and horizontal case; these effects can easily make this analysis unreliable at the ns scale, where a complete simulation after place-and-route is necessary).

Array	Pixel pitch [µm]	Size [pixels]	Size [mm^2]	v_S [ns/cm]	v_H [ns/cm]	min clk [ns]
$\mathcal{A}$	50	512×512	25.6×25.6	7	26	36
$\mathcal{B}$	50	128×2048	6.4×102.4	7	10	9
$\mathcal{C}$	10	128×2048	1.28×20.48	7	26	2
$\mathcal{R}$	50	16×256	0.8×12.8	7	26	1

Table 5.2: Timing performances for the different matrices, calculated as explained in the text.

As a reference a clock period of 12 ns (83 MHz frequency) was taken because it is plausible according to other devices developed in a similar technology (ARCADIA-MD1). This clock works with all the arrays discussed but $\mathcal{A}$, which is used in simulation only as a reference implementation of a square hash array[6]. A complete analysis studying whether the architecture can withstand such clock is planned after the place-and-route. According to Table 5.2, faster clocks should be possible; hence, the value chosen here is conservative.

As a safety net for the synchronous logic, all the state machines are capable of returning to their idle state if there are signals not timed correctly: lost data are preferred to fake hits.

5.2.2 Power performances

In order to obtain a reliable power consumption estimation for the digital architecture, it is necessary to take into account three contributions: the functioning of pixels, the communication of information from the pixel to the periphery, and the functioning of the periphery. The first two components are estimated quantitatively in this work, the last one is not as its value relies too heavily on the place-and-route phase, a level of implementation not achieved yet for this project; only some generic concepts are outlined in the next paragraphs. These estimations have been carried out by either direct calculations looking at the technological library or by loading such a library into a simulation software (Modelsim [131] was used). In the next paragraphs, these contributions will be evaluated and an overall estimate of the power consumption for the matrix $\mathcal{R}$, the one designed as described in Section 5.1, is given. An interpretation of this result is found later in Section 6.

[6]Also other assumptions, like the fact that the signal travel time depends only on the region height in the Monte Carlo simulations, fails when studying this large square array.

Pixel power consumption

The digital component of the pixel region consumes energy when it is powered on (static consumption), whenever there is a new configuration, there is a new hit, the hash code is changed, or the hit is erased from the buffer. Both the configuration and new hit management is really simple, as the only action performed is either a set or reset of a latch. This is an advantage in terms of power (and space as well) with respect to many other MAPS like ALPIDE, which need full encoding and decoding in-pixel logic [120]. In terms of hash code modification, on the other hand, the signal is just connected to an `OR` gate, part of the chain that reaches the periphery. Given these premises, the pixel functioning digital power consumption is expected to be negligible in comparison with all the other contributions.

The calculations have been performed on two different possible implementations of the pixel: one as discussed in Section 5.1.1, and an alternative obtained from the first one by removing the gates used for the pixel region configuration (avtually shown in Figure 5.6). The results are summarised in Table 5.3 for dynamic power consumption and in Table 5.4 for leakage power.

Operation	Full pixel region	No-configuration
Hit buffering	8.5×10^{-2} pJ	6.0×10^{-2} pJ
Hit reset	7.5×10^{-2} pJ	8.0×10^{-2} pJ
Readout-and-reset	8.5×10^{-2} pJ	9.0×10^{-2} pJ
Buffereing, readout, and reset	17.0×10^{-2} pJ	15.0×10^{-2} pJ
Bit configuration	5×10^{-2} pJ	–

Table 5.3: Dynamic power consumption in a pixel region of the implemented array $\mathcal{R}$. Both complete pixel regions and pixel regions without configuration have been considered. The propagation of the hash signal was not taken into account, as its contribution is calculated separately.

	Full pixel region	No-configuration
Leakage	66 nW	28 nW

Table 5.4: Simulation of a pixel region leakage power in the implemented array $\mathcal{R}$. Both complete pixel regions and pixel regions without configuration have been considered.

Communication

The hash lanes and the data lanes behave differently in terms of timing performance; the same holds for power consumption. To estimate the power performance, the same technological library used for the delay estimation in the most pessimistic corner (this time from the

power point of view) is used. The average energy consumption for a bit swap in a data bus is calculated to be:

$$E_S = 2 \text{ pJ/cm bit}$$

Regarding the hash signal communication, it consumes more power as there are more gates. The calculated values E_H are reported in Table 5.5, as this figure depends on the geometry of the sensor.

Array	Pixel pitch [µm]	Size [pixels]	Size [mm^2]	E_S [pj/cm bit]	E_H [pJ/cm bit]	E readout [pJ]
$\mathcal{A}$	50	512×512	25.6×25.6	2	6	46
$\mathcal{B}$	50	128×2048	6.4×102.4	2	4	140
$\mathcal{C}$	10	128×2048	1.28×20.48	2	7	43
$\mathcal{R}$	50	16×256	0.8×12.8	2	4	17

Table 5.5: Communication power performances for the different arrays.

These numbers were used to calculate the average energy needed for a readout, calculated considering that there are 4 hash bits (P_X, P_Y, P_U, and P_V), and 5 signal bits (`READ`, `CLEAR`, `ROWSEL` from the periphery to the pixel, and the response of the pixel (`PID` and `SYNC`), taking into account, of course, also the height of the array.

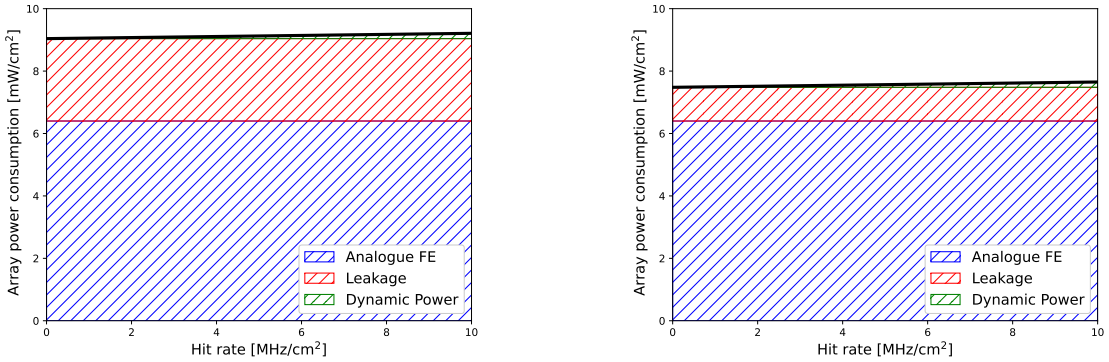
This method of evaluating communication power consumption can be found in the literature and is confirmed experimentally: in the case of the ALPIDE sensor, the power consumption due to communication alone was estimated to be 72 pJ per hit [120], while the measurements revealed 100 pJ per hit transmission [54]. The values calculated in this subsection have been confirmed through Monte Carlo simulations: by using ModelSim to inject signals in random positions in chains of `OR` gates or buffers, the agreement was within 10% of the calculated value.

Periphery

The periphery is a very crowded region of the design and its power consumption estimation would need a complete place-and-route to be reliable. Considering that the development did not reach this stage, no estimation was performed. Anyway, what is implemented in this project does not differ much from what is implemented in other MAPS detectors: the only major difference lies in the necessity for the hash decoding operation that must be performed to generate the valid addresses. Considering that it is not an operation particularly computationally demanding, there is no reason to expect performance far from the other existing MAPS from this point of view. The quantitative estimation on the power consumption has not been carried out on this part of the sensor as the uncertainties without place-and-route are expected to exceed by far the 10%.

Overall power consumption

These contributions have been combined in order to have an estimate for the power consumption in the bulk of the matrix; i.e., in the region occupied by the pixels themselves (thus neglecting the periphery). This power consumption, including its dynamic contribution, depends on the hit rate, and the results can be seen in Figure 5.26a both for the implemented design and the one without configuration, showing how a power below $10\,\mathrm{mW\,cm^{-2}}$ in the sensor bulk being, actually, feasible. For completeness, in that figure, also analogue front-end consumption is reported. This value is borrowed from ALPIDE [29], [120], as this thesis did not touch on optimisation of the analogue front-end. The overall power consumption in the array is dominated by this analogue front-end power consumption. The second most important contribution is the leakage power, which can be reduced by optimising the configuration (more than half of the leakage is due to transistors present to implement the correct configuration of the chip). The dynamic contribution is heavily suppressed by the fact that the design is idle most fo the time thanks to the data sparsification.



(a) Full pixels design.

(b) Pixels without configuration.

Figure 5.26: Estimation of the power consumption in the array that implements full pixels or without configuration.

5.3 Monte Carlo simulations

To quantify how the proposed design performs in a realistic physical environment, as well as stress the design, different simulations have been run. The approach pursued starts from very simple studies of the mathematical properties shown in Section 4.3 (Section 5.3.1); to more precise ones taking into account all the timing delays present (Section 5.3.2), and the most complete ones which take into account the actual physical implementation of the architecture (Section 5.3.3).

Monte Carlo simulations can be performed by using many different programming languages, chosen according to the complexity of the issue under study:

- Interpreted scripts written in a language like Python [104], Javascript [132], Matlab [133], Wolfram [134] or similar. Relatively easy to write, fast to run, not ideal for

complex structures and calculations with an original approach;
- Compiled programming languages like C [135], C++ [102], C# [136], Java [137] or similar. Although these programs are usually longer to develop, by using these tools it is easier to completely understand the operations performed inside the simulation and create a custom framework for the work;
- Hardware-oriented programming languages like SystemVerilog [99] or VHDL [138] that can perform simulation on the actual digital architecture implemented in the very same language. Realistic and complete, but quite long to run;
- Hardware-specific simulation after place-and-route. These simulations are actually the most reliable ones, as they take into account physical effects such as the delays due to the wire length or the parasitic capacitance between wires, but are very specific and technology dependant, and need a design after place-and-route.

In this work the first three techniques have been used. Python scripts were used to understand how different projections or array sizes can affect the pixel reconstruction capability, a C++ software has been used to study the efficiency of the readout architecture as a function of the particle rate with relatively high statistics, and a SystemVerilog testbench has been used to test the stability of the architecture from a timing point of view, as well as validate the C++ software, and extract the power consumption of the pixel region.

5.3.1 Mathematical properties

The goal of the first set of simulations is to verify some statements demonstrated mathematically in Section 4.3, as well as quantify how better it is to use optimal sets of projections, and the fraction of hits that starts clogging the readout architecture. All these simulations start from the same initial condition: an array of addresses is created and filled it with a fixed number of hits, randomly chosen among all the pixels still empty. This object is called a **frame**. When a frame is populated with a fixed number of data, all valid addresses are calculated by decoding the hash function produced by the set of hit pixels; a software approach allows to easily calculate all the valid addresses. Then, the number of valid addresses is compared with the actual hit population in the array, and some performance figures are calculated. These operations are repeated for a fixed number of frames, in this case 6.4×10^4, and the results are averaged. The following figures of merit have been estimated:

- **Rate of wrong reconstructions**, defined as the ratio between the ghost addresses and the correct addresses (using the nomenclature already introduced in Section 4.2). This number gets higher as the number of hits per frame gets higher, as ghost addresses are progressively more probable, and its increase implies a worse-performing device;
- **Perfect set reconstructions**, defined as the fraction of frames that can be reconstructed perfectly. A frame is reconstructed perfectly when the only addresses reconstructed are the correct ones. In this case, obviously, a higher percentage corresponds to a better performing device.
- **Rate of wrong guesses**, defined as the number of ghost readouts out of the correct ones. This figure is more physics-driven: starting from the set of valid addresses,

the first address (the leftmost between the downmost, the same ordering implemented physically) is identified as either correct or ghost. If it is correct, this hit is removed, the hash code is calculated once again, and this procedure is repeated; if it is a ghost, then the next address (in order) is tested. The figure of merit is the ratio between the times the address read was a ghost and the times it was actually a correct address. This simulation traces the working principle of the hash decoder, counting how many readouts are necessary before emptying the array. In this case, the higher the rate of wrong guesses, the worse it is in terms of performance.

These simulations have been run on five different hash architectures:

- $\mathcal{M}_{128}$: a 128×128 array using three optimal projections:

$$P_X = (1,0) \qquad P_Y = (0,1) \qquad P_U = (1,1) \tag{5.12}$$

- ${}^{s}\mathcal{M}_{128}$: a 128×128 array using three projections: two optimal ones and a third suboptimal[7] with respect to one of the others:

$$P_X = (1,0) \qquad P_U = (1,1) \qquad P_V = (1,-1) \tag{5.13}$$

- ${}^{s}_{4}\mathcal{M}_{128}$: a 128×128 array using four projections: three optimal ones and a fourth suboptimal with respect to one of the others:

$$P_X = (1,0) \qquad P_Y = (0,1) \qquad P_U = (1,1) \qquad P_V = (1,-1) \tag{5.14}$$

- $\mathcal{M}_{127}$: a 127×127 array using three optimal projections:

$$P_X = (1,0) \qquad P_Y = (0,1) \qquad P_U = (1,1) \tag{5.15}$$

- ${}_{4}\mathcal{M}_{127}$: a 127×127 array using four optimal projections:

$$P_X = (1,0) \qquad P_Y = (0,1) \qquad P_U = (1,1) \qquad P_V = (1,-1) \tag{5.16}$$

This set of architectures has been chosen to allow to measure how the addition of a projection improves the sensor performance, validate how optimal sets of projections are better than suboptimal ones, and quantify the overall performances of the hash approach in terms of wrong reconstructions. For these mathematical-driven simulations, a ≈ 128-sized array has been chosen as it would be too long to simulate larger matrices, and this size is enough to demonstrate the mathematical properties. Two different array sizes are used (side 128 and 127) in order to compare even-sized matrices and prime-sized matrices[8].

The results of these three simulations are shown in Figure 5.27, with a number of hits per frame ranging from 2 to 25 (i.e., a population fraction at most of 0.15% of the total number of addresses). A first result confirmed was expected: the more projections are used in the architecture, the better the device performs. Concerning the choice of projections, optimal

[7] Also in this section, suboptimal projection must always be interpreted as **suboptimal but not trivial** projection.

[8] No sets of four optimal projections exist in even-sized matrices, as demonstrated in Section 4.3.4.

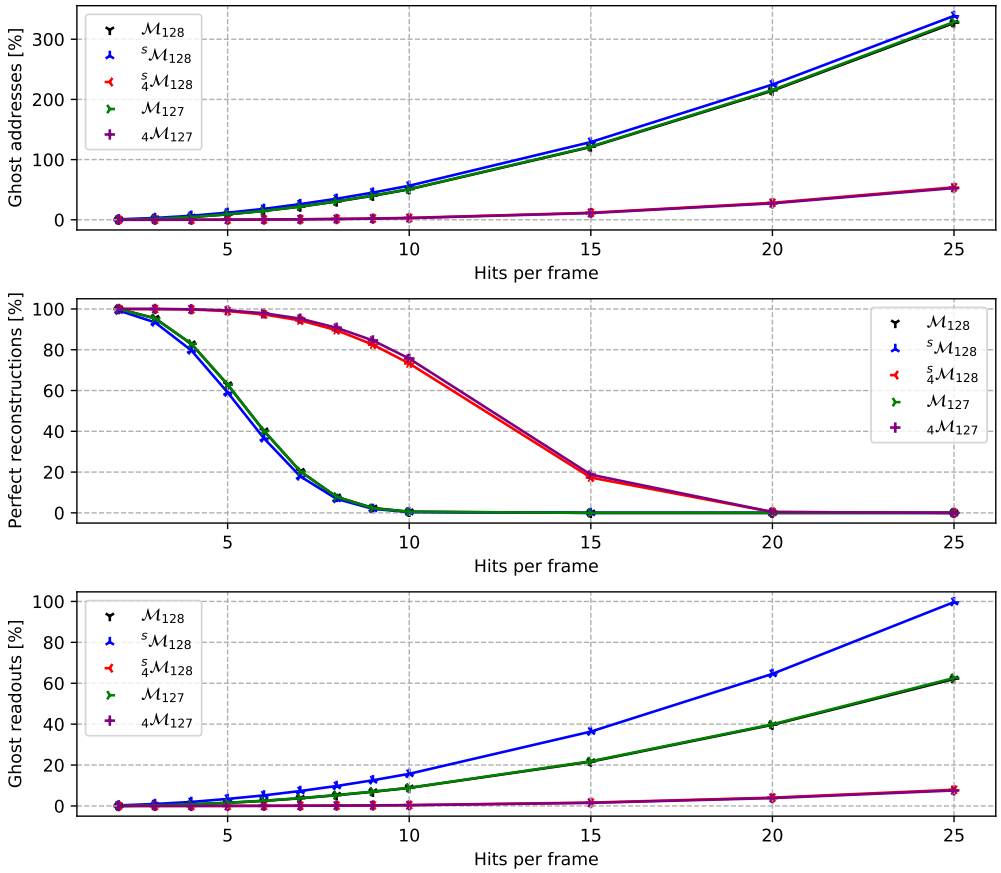


Figure 5.27: Rates obtained from the first three simulations. In first figure, rate of wrong reconstructions (the percentage of ghost addresses); in the second figure the rate of perfect frame reconstructions (reconstructed pixels equal to actually populated pixels); in the third figure the percentage of ghost reads trying to empty the array. All 5 architectures compared.

sets of projections are always preferred in comparison with suboptimal sets of projections (e.g., the blue curve behaves systematically worse than the black one, despite being similar). Another effect that can be seen is that the size of the array changes the performances only if second order effects are considered (this can be seen by looking at the black curves and the green ones, that are superimposed in all the plots shown, as $\mathcal{M}_{128}$ and $\mathcal{M}_{127}$ implement both a set of three optimal projections). Moreover, from this plot it is clear that the addition of a fourth projection can drastically improve the performance of the detector even if such projection makes the set suboptimal. In fact the difference between the purple and the red curve in the plot, representing respectively an optimal and a suboptimal set of four projections, is not so remarkable, even if statistically significant (in this plots uncertainty, although present, cannot be seen behind the markers).

These simulations also show the architecture behaviour in terms of many-hits clogging: the benchmark discussed in Section 4.1 is a 5% increase in time and power consumption in comparison with the optimal conditions. Considering that every ghost readout takes time (and power) comparable with a good readout, this constraint can be moved to a number of

ghost readouts of 5%. This is reached for the architectures implementing three projections at ≈ 7 hits and for the architectures implementing four projections at ≈ 20 hits: when the average population is below, many-hits clogging is not a major problem. For four projections, this corresponds to a fraction of hits in the array around 1 per mille, in perfect agreement with the sparsification hypothesis assumed.

5.3.2 Efficiency simulations varying the particle rate

This readout architecture might be useful in several contexts, as those introduced in Chapter 2. In some of these applications (as is outer space, where the power consumption constraint would make such an architecture interesting), particles to be detected reach the tracker at random moments and there is not any external trigger. In this condition, the time between an event and the following one is an exponentially distributed random variable:

$$f(t;\tau) = \frac{1}{\tau} e^{-\frac{t}{\tau}} \tag{5.17}$$

with τ the parameter representing the average time between a hit and the following one. Regarding spatial distribution, here a spatially uniform particle source is considered. τ parameter is combined with the detector dimensions to use as the independent variable the **hit rate**, measured in $\mathrm{Hz\,cm^{-2}}$. The simulation discussed in this section is more realistic than the previous ones as it introduces the **time** variable, therefore the timing performance of the device under study must be taken into account, as discussed in Section 5.2.1.

Concerning the cluster size, for the simulations reported here a cluster size equal to one has been chosen. This has been done to disentangle as much as possible the readout architecture from the actual physical device. Physically, the cluster size can depend on:

- The array pitch: the smaller is the pitch, the bigger is the average cluster size;
- The array thickness: the thicker is the sensor, the bigger is the average cluster size, as the generated charges can diffuse to neighbouring pixels;
- The energy released by the particles: the more energy is deposited in the sensor, the higher is the cluster size, as there are more charges collected;
- The threshold set in the detector: the higher the threshold set, the smaller is the average cluster size, as more electrons are necessary to actually trigger a hit;
- The impinging angle of the particle, as the more perpendicular the impinging particle is, the smaller the average cluster size, as charges are produced as less horizontally spread.

By using the **hit rate** instead of the **particle rate** it is possible to not include assumptions on any of these quantities to study the performances of this architecture. Another advantage of the hit rate approach consists in the fact that these simulations remain valid in the case of a clustering algorithm: in this case a hit correspond to an arbitrary number of hits in a single super-pixel.

This section presents both the software and the results; some comments on the latter are reported as well. An example of a debug run of the software can be found in Appendix C,

and a more specific comment on the results in comparison with the current state-of-the-art is discussed in Chapter 6. The description of the programme is actually simplified as this document is not meant to be too technical.

The reference array was already introduced in the last section as $\mathcal{A}$ (512×512 square array with a pitch of 50 µm) implementing a suboptimal set of four projections: P_X, P_Y, P_U and P_V (same projections set discussed and implemented in Section 5.1, but no clustering and different dimensions, as the hash sector in the physically implemented architecture is smaller). Other geometries (together with alternative architecture implementation with in-pixel memory and different beams) are discussed later.

In such an architecture, time is consumed in the different stages of the readout algorithm. The following processes are modelled as time-consuming, taking as a reference the discussion introduced for general clockless readouts:

- **Hash communication**: when a particle hits a pixel, the hash signal takes some time to reach the end of the column, as it needs to traverse the whole hash chain. The signal speed is considered fixed; i.e., the time needed is linearly dependant on the height of the pixel. Horizontal delay is neglected. The same time is lost the other way around after a reset (more details in Section 5.2.1);
- **Address guessing**: the periphery needs some time to come up with an address given the hash code. This process is synchronous to the clock used by the whole periphery. Guessing can be done in different ways: in the current implementation, the approach described in Section 5.1.5 is replicated. The time it takes to generate a valid address depends both on the clock period and on the current hash code (in first approximation, on the number of hits in the array).
- **Readout-and-reset request**: the end of column sends the readout request along the correct column. This signal takes some time to reach the correct pixel, and this time depends on the vertical position of the pixel itself, just like the hash communication stage. Slightly after the readout, a reset request is sent as well (see Figure 5.8). These two signals travel faster than the hash signals, as discussed in Section 5.2.1.
- **Data arrival**: data must travel along the column on a line parallel to the readout request line, and of course it takes some time to reach the end of the column. The speed of the signal is considered to be equal to the speed the other way around; hence it is the same as the speeds of the readout request and the reset.

From a technical point of view, this simulation takes advantage of the power of the **Object-Oriented Programming** (OOP) approach, in particular by using the programming language C++. It is organised in two different active objects, performing the proper actions when triggered at the right simulation time. These two objects are **Entities** and **Events**. This software has been validated by comparison with the SystemVerilog simulation (see Section 5.3.3). Although a complete SystemVerilog simulation would be more precise, this C++ alternative allows a study with higher-statistics.

Entities represent an object in the actual physical system (the array, the periphery, and the beam), while events represent an instantaneous occurrence of something happening somewhere in the device (like the broadcasting of an electric signal, or a signal reaching its

destination). Events are loaded in a software structure called the **stack**. It is a container that keeps track of the events that still need to happen, ordered according to their happening time. Whenever an event is created, it is added to the stack with a corresponding *pop-time*. Events are popped out of the stack one by one when the simulation clock reaches their pop-time and only when they are the event with the lowest pop-time inside the stack. Once an event is popped, its actions are performed; e.g., entities are updated or other events are added to the stack.

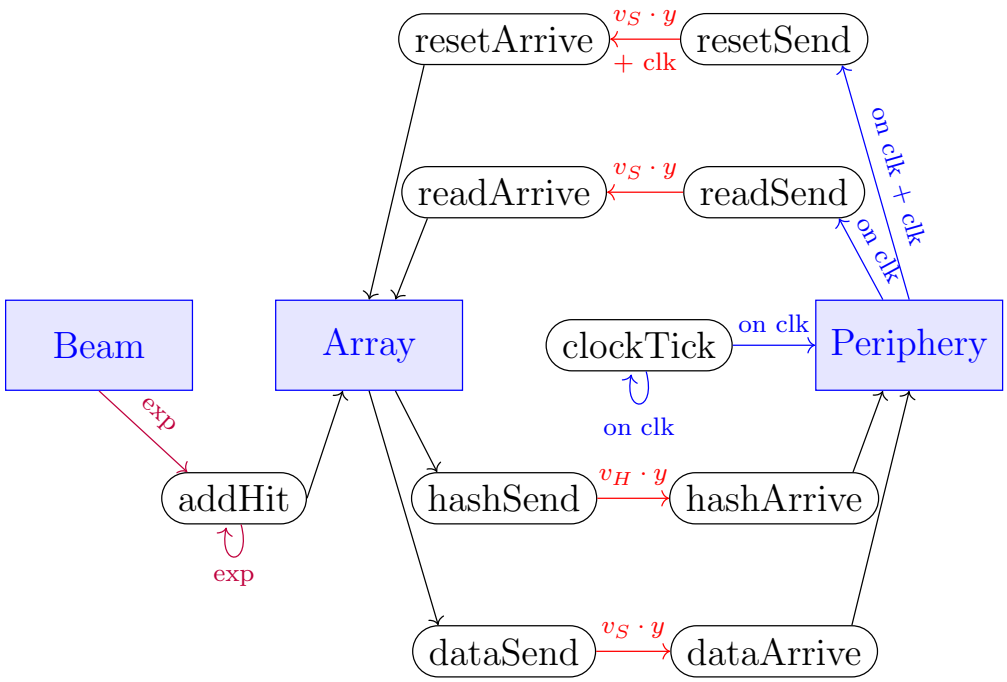


Figure 5.28: Simulation structure. Inside rounded rectangles the events are listed, while in boxes the entities. An arrow is to be read as "That creates" when it points to an event, or "that updates" when it points to an entity. Black arrows show instantaneous event creations, purple arrows show events delayed using a random number generator, red arrows show events delayed depending on the pixel physical position in the array and blue arrows show events synchronous with the periphery clock.

In Figure 5.28 a schematic representation of all the entities and the events is shown together with their relationship. In this section, v_S and v_H represent the data travelling speed that can be found in Table 5.2, and y is the height of the pixel currently involved; they are reported for clarity in few pages.

The entities

- **Beam**: the beam is a completely asynchronous entity, as it is by definition not synchronised with anything. The beam is responsible for starting the simulation with the first particle, creating the first **addHit** event. Beam actually implements the time randomisation through Equation 5.17. By changing the beam entity this simulation can be validated by using previously set-up conditions instead of randomised one. Several

beam implementations are available, here the uniform one is used, a Gaussian one is presented later.

- **Array**: The array can be thought of as the set of all the pixels, each one with its own address and state (either full or empty). Whenever the array updates its status (because a hit has been added or removed) it triggers the **hashUpdate** event. Whenever a readout request arrives, data are sent towards the periphery; whenever a reset request arrives, data is erased and hash is updated again. Sometimes both readout-and-reset are called on a pixel that contains no real hit, in that case the hash code is not updated. Also the array is completely asynchronous.
- **Periphery**: The periphery actually implements the readout scheme as implemented and described in Section 5.1.5 and Section 5.1.2. It is completely synchronous, as every action is performed only on a clock tick. Here the RTL logic is replicated in C++. In the periphery entity, every time the hash code changes the list of addresses is calculated. Each address takes some time for its calculation, depending on the hash code itself. When a valid address has been calculated, the readout-and-reset requests are sent toward the array and this last address is saved. This operation is necessary to avoid asking multiple times the readout of a ghost address. After a readout, the periphery remains dead for a clock cycle to prevent collisions. The periphery has been written in the most general way possible, making it easy to simulate in the future a different number or type of projections or alternative decoding implementations.

The events

- **addHit**: the analogue architecture is giving a signal and a hit appears in the array. This event is continuously created by the beam and interacts only with the array. After an addHit event is popped, a random time is generated again using Equation 5.17 and the following addHit is loaded in the simulation. This process stops only when a predefined number of hits has been simulated and simulation is over;
- **hashSend**: the hash code changed because the status of the array changed; i.e., a new hit is in the array or a hit has just been removed. This event is prompt after the **addHit** and the **resetArrive** and is created by the array. When popped, it creates the event **hashArrive**;
- **hashArrive**: the hash code modification reached the periphery and now there may be a different hash code[9]. This event is created by the **hashSend** event and changes the periphery status. This event is delayed, as its pop-time is increased with respect to its creation time by:

$$t_H = v_H \cdot y$$

- **readSend**: a readout request is sent from the periphery toward the array. This event can happen only on a clock tick, and the number of clock cycles needed between the hash modification (or the last readout) and this event depends on the hash population. This event is created by the periphery and creates a **readArrive** event;

[9]Sometimes, although the array population changes, hash code does not change, as the corresponding hash bits are high because of other hits in the array. In this peculiar event, the periphery does not see a different hash code. This rare occurrence is taken into account correctly.

- **readArrive**: the readout request has reached the pixel and array is pinged with the readout request. This event is delayed depending on the height of the pixel:

$$t_R = v_S \cdot y$$

  This event interacts with the array;
- **dataSend**: the array responds to the readout request and data start travelling from the array toward the periphery. This event is prompt, and creates a **dataArrive** event. Sometimes this event does not actually send data because readout pixel is empty, it is therefore a ghost. Anyway, the whole logic described in Section 5.1.2 is replicated, therefore the periphery is informed anyway;
- **dataArrive**: data reach the periphery and are collected. This is the end of the readout routine and periphery is updated with the data. Delay time is:

$$t_D = v_S \cdot y$$

- **resetSend**: reset is sent toward the pixel just read. This event is created by the periphery one clock cycle after the readout request. Reset is also sent when hash population is too high: if there are too many events in the array, it is better to reset all pixels instead of trying to read them one by one, as a sensible fraction of the guesses are likely to be wrong. This is explained in more details later;
- **resetArrive**: reset reached the array after the travelling time along the array and data is erased from the array. Delay time is:

$$t_R = v_S \cdot y + \text{clk}$$

  A clock cycle is added to reproduce the behaviour of the SystemVerilog simulation.
- **clockTick**: a clock tick in the periphery. The distance between a clock tick and the following one is fixed, and determined by the clock period. A clock tick asks the periphery if there is any action to be performed, and does not interact with the array, as it is asynchronous.

Timing constants

The behaviour of the architecture clearly depends on the amount time that the various events take; i.e., on the clock period and the propagation speed. These constants have been calculated and discussed in Section 5.2.1. The calculated values are listed in Table 5.2, and reported here for clarity:

Array	v_S [ns/cm]	v_H [ns/cm]	clk period [ns]
$\mathcal{A}$	7	26	12

Reset ratio

This readout architecture relies on the idea that given a hash code, that is a set of bits representing various projections of the hits currently in the array, it is easy to guess the addresses of such hits. This is not always the case: if there are too many hits in the array, the hash code will contain a lot of 1 bits; hence, there will be many possible addresses to try and read, many of which will be ghosts. A more prominent effect is the address production time: according to the algorithm discussed in Section 5.1.5, if there are too many hits there will be many *tentative addresses* that are not valid. A slower readout actually translates to lower efficiency, as pileups become more likely and time to perform a readout increases even more, in a dangerous positive feedback leading to the full congestion of the device. In order to avoid this, the architecture implements an **internal full reset**. The hash decoder actually counts the fraction of high hash bits at every clock cycle. If this fraction is greater than a customisable level, instead of producing one of the (many, likely wrong) addresses for a readout test, a request for reset is broadcast to all the pixels composing the array. This routine leads to a loss of good data, but the amount of data that would be lost if it was not present can be higher. This routine is a safety net that limits data loss when such a readout architecture is outside of its application regime and the balance between the hits created in the array and their readout is lost; it should never be broadcast if the sensor is used in its correct environment (low enough particle rate). When this internal full reset is dispatched, the hash decoder goes to a dead state that does not produce any valid address for some clock cycles (in the current implementation, four), to protect the CLSMs stability and wait for the updated hash code. This feature is not discussed in Section 5.1 for clarity.

Simulations have been performed in order to identify the optimal value of this reset level parameter. Here a study on its effect on the sensor efficiency is reported: a value of 0 means that the array is always reset, a value of 1 means that it is never reset. The unit on the independent axis is the hash reset level fraction; i.e., the fraction of high hash bits with respect to the total number of hash bits. In Figure 5.29 this plot is shown for three different particle rates[10]. It is easy to see how the efficiency strongly depends on this parameter. For low particle rates (up to $1\,\mathrm{MHz\,cm^{-2}}$) the architecture is not stressed as the number of read hits is comparable with the number of new hits and many-hit clogging is not a problem. For higher rates (blue and red points) many-hits clogging becomes a problem and resetting the array properly helps keeping the efficiency high. A more precise estimate on the highest bearable flux can be found in the simulations discussed later.

A few comments regarding this plot can help understanding how this architecture behaves:

- For low reset trigger values the periphery decides to reset the array before actually trying to read the hits: hits gets erased well before clogging is actually a problem, resulting in a highly inefficient device;
- For high reset trigger values the periphery seldom asks for full array reset, and readout

[10]This plot, like all the following ones in this section, have the error bars, but the statistic is high enough for them to be covered by the markers

is inefficient. In this situation, the readout becomes slow, and this leads to many pileup hits, hence a still highly inefficient device;

- There is a maximum at 1.5% reset level. This actually sets the bottleneck for the readout, that is not due to the presence of many ghost address but rather to the amount of time that the hash decoder takes to produce a valid address. In fact, as explained in Section 5.1.2, there are at least four cycles between a readout and the next one, and the number of cycles needed to produce a valid address depends on the hit population m and is calculated in Section 5.1.5. On average $m/2$ clock cycles are needed to find the first valid address. This means that if the hit population is 8 (and the number of high hash bits is around 32, that is, below 1. 6% of the 2048 hash bits[11]) or below the decoder will likely already have a new address when the new readout is possible, if this is not the case some more time will be lost to generate the next valid address, and the time needed for each subsequent readout will grow more and more.

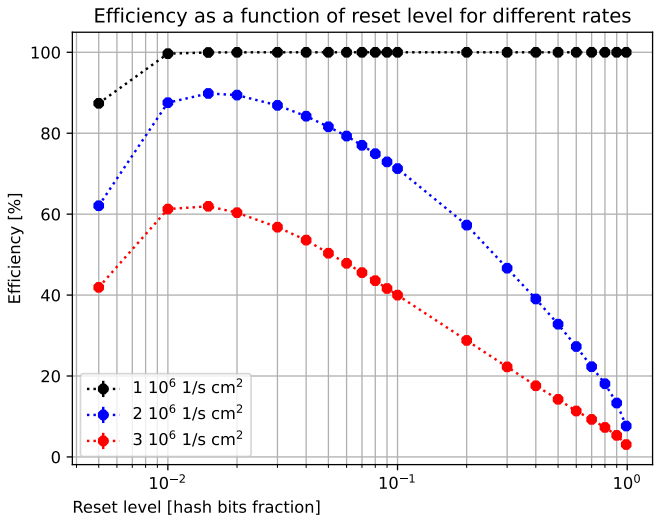


Figure 5.29: Efficiency for the hash architecture versus the level for the full reset. In the three colours, three different particle rates.

Rate capability

The main goal of this study is an estimation of the efficiency as a function of the hit rate. This simulation was performed in a way similar to the previous one, but although in the previous simulation the hit rate was kept constant varying the reset fraction, in this case the reset fraction was set to its optimal value (1.5% of the total hash width), and the hit rate was varied. Also here, a million particles were generated for each rate value. The results can be seen in Figure 5.30.

[11] Here there are 512 bits for each of the 4 hash words. Some more precise information on the probability to have more pixels on the same hash line can be found in Section 5.1.5.

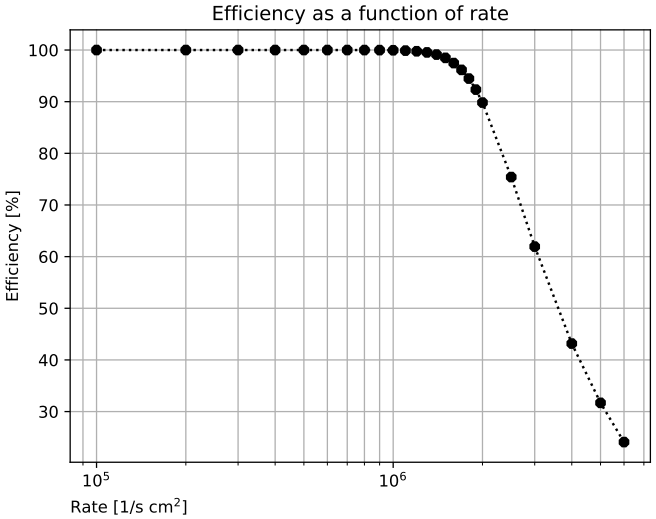


Figure 5.30: Efficiency for the hash architecture as a function of the hit rate. The reset rate is held constant. Inefficiencies can be due to full array resets (lost hits) or to hits in an already populated pixel (pileup hits).

The figure clearly shows that efficiency is high (close to 100%) with particle fluxes up to $1\,\mathrm{MHz\,cm^{-2}}$ and then it starts dropping, reaching 95% at $\approx 1.75\,\mathrm{MHz\,cm^{-2}}$. This is a first estimation of the architecture performance in the case of a square array. From this plot it is possible to confirm that the reset rate, although deeply modifying the sensor performance, is nothing but a safety net: the drop in the efficiency is due only to the onset of the whole matrix reset.

Alternative Matrices and simulations

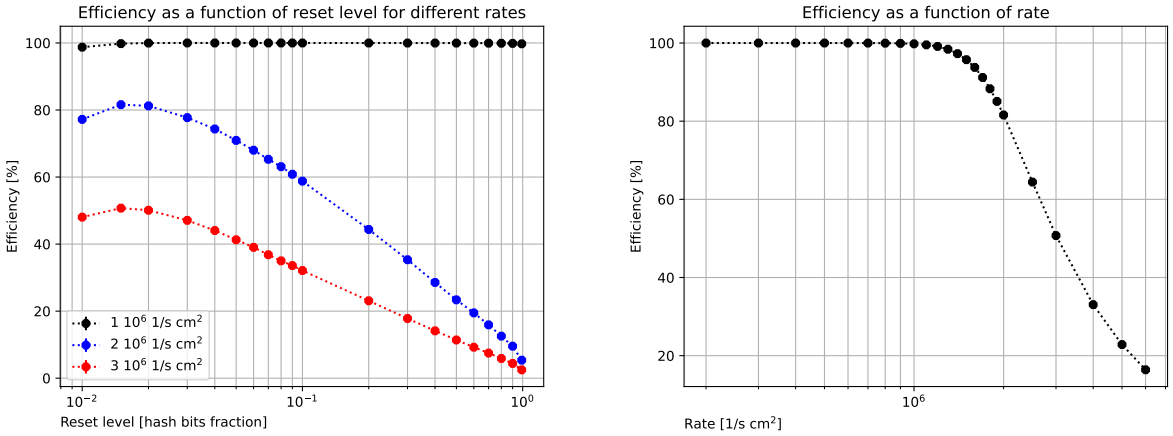
The simulation framework presented allows to explore the possibility of alternative matrices, and set performance remarks for different beams. First, three rectangular devices are studied; then in-pixel memory is added to the array to try to improve the readout capability, and finally the performances of the sensor using a gaussian beam is investigated. In the last subsection, multi-hits clusters effect is evaluated.

Different geometries

Although the square is the natural shape for the hash architecture, a higher aspect ratio rectangular layout (tall and slim) is advantageous when building large area sensors, as the whole pixel array can be built by tiling identical, independent sectors along one dimension (as discussed in Section 5.1). This approach has been followed also in the implementation developed in this thesis, discussed in Section 5.1. A rectangular implementation of the hash architecture would therefore be particularly appealing for the realisation of very large area sensors; as the architecture readout is barely parallelisable, by splitting the overall area into

independent sectors it is possible to achieve faster readouts and simplified design layout. Here, the four geometries reported in Table 5.2 have been fully simulated, and for each one the specific timing characteristics have been used. The clock used was always 83 MHz in all cases.

Concerning the array $\mathcal{B}$, the study for the optimal reset value can be seen in Figure 5.31a, while the rate capability can be seen in Figure 5.31b. The reset level selected for the rate study was yet 1.5%, as the selector bottleneck remains the same (this array has the same number of pixels of $\mathcal{A}$; thus, mathematical properties are the same). By looking at Figure 5.31b in comparison with Figure 5.30 it is possible to see that, as it was expected, the rate capability is lessened if the the aspect ratio is increased; nevertheless, the results remain good.



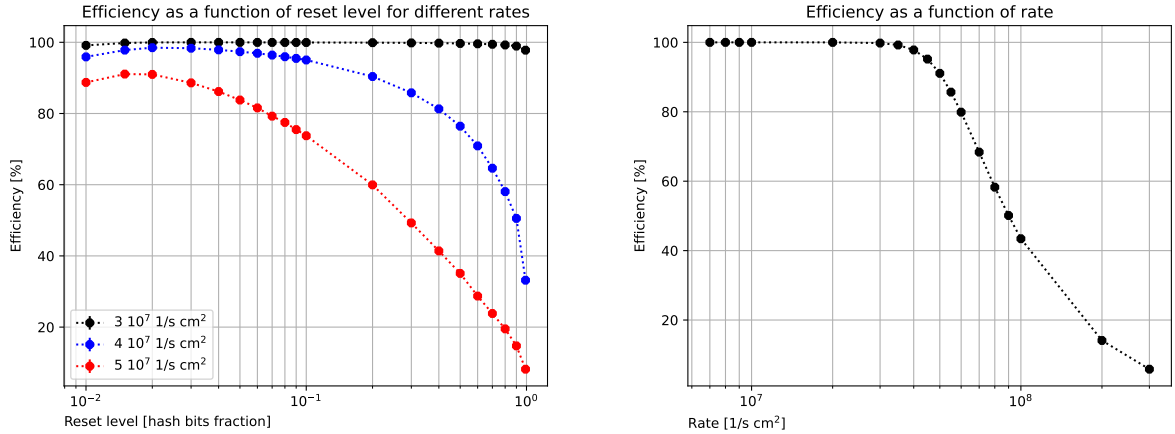
(a) Study on the reset leevl. **(b)** Efficiency study.

Figure 5.31: Simulations on array $\mathcal{B}$ (rectangular, 128×2048 50 µm pitch pixels).

Contrarily, the results are quite different for array $\mathcal{C}$, where pixels are smaller. In Figure 5.32a the reset rate is studied (here the rate scale is different in comparison to previous plots). The optimal rate is nevertheless of the order of 1.5%, justifying once more the explanation given. This result has been used to study rate capability, showed in Figure 5.32b. In this implementation, hit rate capability is far higher (efficiency is 99.8% at 30 MHz cm^{-2}). The better rate performance is due to how data are presented: in this array, a given particle flux given in MHz cm^{-2} corresponds to a lower value in terms of hits per pixel, as pixel density is much higher.

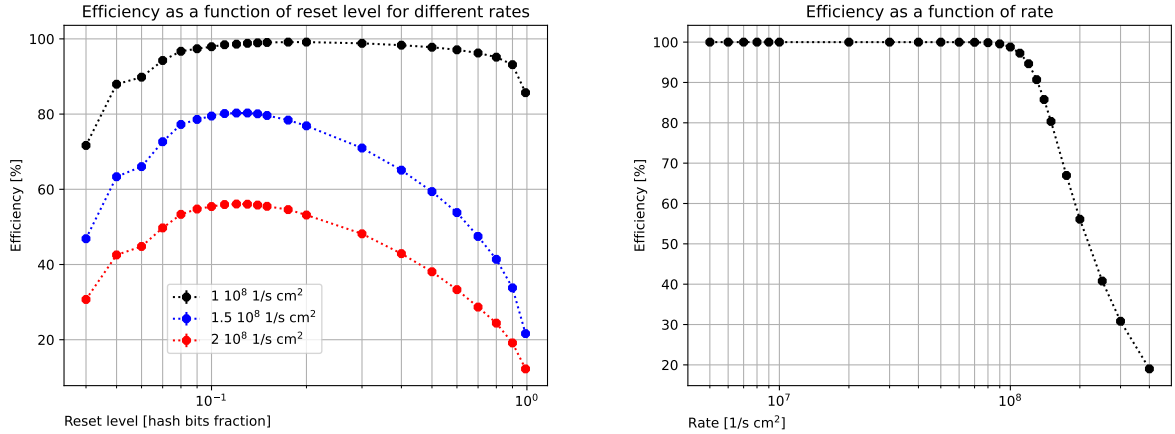
For what concerns array $\mathcal{R}$, the study performed here retraces the other simulated arrays: with a standard flavour and no clustering implementation. The addressing and the readout-and-reset procedures remains the same from a timing point of view. This simulation actually depicts the performance if the 50 µm-pitched pixel-regions were actually simple pixels with the same size. The study on the reset level is presented in Figure 5.33a, while the study varying rate is presented in Figure 5.33b.

This array is interesting as it implements a smaller hash side in comparison with all the other matrices simulated so far; thus the mathematical properties of this array are different.



(a) Study on the reset level. **(b)** Efficiency study.

Figure 5.32: Simulations on array $\mathcal{C}$ (rectangular, 128×2048 10 µm pitch pixels).



(a) Study on the reset level. **(b)** Efficiency study.

Figure 5.33: The results of the simulation on array $\mathcal{R}$ (rectangular, 16×256 50 µm pitch pixels).

One of the most clear differences in comparison to the other simulations is the rate reference level: the rate capability of this sensor, being much smaller than the others, is higher. At $9.7 \times 10^8\,\mathrm{MHz\,cm^{-2}}$ the array efficiency is still above 99.5%. This is due to the concurrence of the fact that there are far less pixels for the periphery to control, and readout-and-reset is faster, as there is less time spent reaching the correct pixel. By looking at the reset figure the hypothesis on the reason for the optimal reset level can be confirmed: applying Equation 5.8 the result is that 8 hits generate 30 high hash bits; in this array size this corresponds to ≈12% of the total hash bits, and this is the maximum in Figure 5.33a). For higher hit populations, the periphery becomes inefficient in finding the valid address among the tentative ones. According to these calculations, the sparsification hypothesis looks not enough for the optimal regime of this architecture: 8 hits in the array corresponds to a population around 2 per mille, while the hypothesis that was set was of 1 hit per mille

pixels. Nevertheless, the maximum flux is high enough for several applications, like those introduced in Chapter 2.

Array with buffers

A possibility that can be explored, depending on the actual physical implementation of the array, is the use of multiple buffers in pixel. In the implementation discussed so far only one latch was present for each pixel to buffer the information of the passing particle; multiple latches may be used to reduce the number of pileups. To study the effect of this possible implementation, a simulation was run by allowing two different hits to be latched inside a given pixel. The squared array taken as reference for simulations $\mathcal{A}$ was studied. Although the number of pileups is actually reduced by the presence of multiple memories, in this architecture:

- Many-hit clogging problems hold all the same: even if multiple data are saved inside the pixel, one information is enough to drive the corresponding hash lines and obtain the hash code;
- Timing performance is worsened: in this implementation the timestamp is associated whenever the address is tested by the periphery, as the array is clockless. If there is multiple information in one pixel only, these pixels share the very same timing information.

The addition of these buffers, according to Monte Carlo simulations, provide little to no advantage in a uniform beam. Results are not actually presented as they lay within the statistical fluctuations of what has already been shown: the reset plot is just like Figure 5.29 and the rate plot is like Figure 5.30. Given these premises, the focus was put on the study of other aspects of the device validation, leaving this possibility as an eventual future upgrade that currently does not show a clear advantage.

Gaussian beams

A completely uniform beam (in both time and space) can be used to study the overall behaviour of the sensor detector, and can well mimic what actually happens when such a chip is installed in outer space or when the irradiation condition is such that the sensor is much smaller than the larger beam spot. However, in some conditions, this premise is not valid, such as when the detector is exposed to beam spots smaller than 1 cm, like in medical physics and nuclear physics applications.

The behaviour of the array (in its standard flavour $\mathcal{A}$: square and with 50 µm pitch) was simulated using a circular Gaussian beam: uniform in time but spatially distributed as:

$$P(r,\theta) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} e^{-\frac{1}{2}\left(\frac{r}{\sigma}\right)^2} \tag{5.18}$$

that is a half-normal distribution (a normal distribution folded to produce values only for positive r).

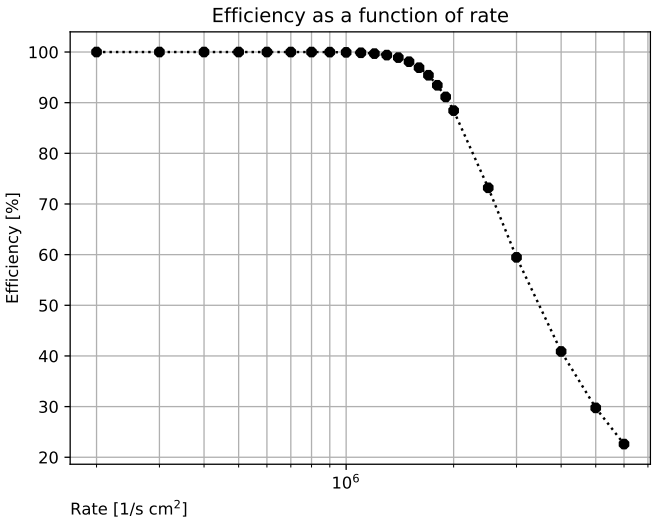


Figure 5.34: Efficiency for the hash architecture using a Gaussian beam centered in the array center.

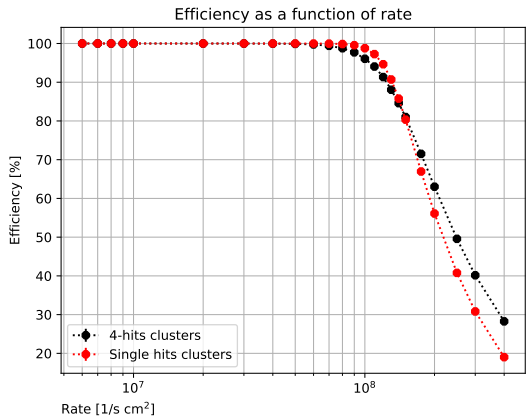
The Monte Carlo software described in this section was run by using such a distribution with the parameter $\sigma = 0.5\,\text{cm}$ still keeping a reset value equal to 1.5% (this optimal value is not expected to change because of modifications the incoming beam distribution). Results can be seen in Figure 5.34.

The architecture performances are almost the same, showing good efficiency also in the case of Gaussian beams. This is not surprising: if the set of projections used were optimal, any position in the array would be independent, and limiting the particle passage to a region would not change by any means the data flow routine.

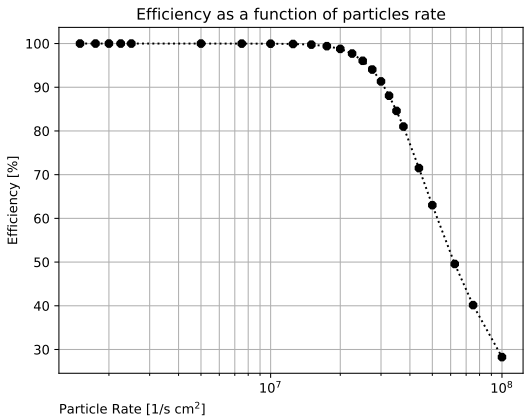
Multi-hits clusters

Another study was performed by passing from the hit-rate to a particle-like spectrum. For this simulation, for every position generated in the array, four hits have been added to the array, randomly orienting a 2×2 cluster. Then, the efficiency of the sensor simulated was normalised to the hit rate instead of the particle rate. The simulation was performed on array $\mathcal{R}$ and the results can be seen in Figure 5.35.

Figure 5.35a can be used to justify the hit-rate approach used here: in the region where the efficiency is high, the two figures are close one to the other. When the efficiency starts dropping, on the other hand, the trend is significantly different: this is simply due to the fact that the reset value set for the 4-hits clusters is different, as the efficiency maximum has been found as located elsewhere. This difference can be explained by the different environment for this simulation: the birthday theorem, used to estimate the maximum hit population in the array equal to 8 for optimal decoding, fails in this case, as the hits are not randomly distributed in this case but placed in a square.



(a) A comparison between 4-hits clusters and single-hit clusters.



(b) Efficiency as a function of particle flux assuming a cluster size of 4 pixel-regions.

Figure 5.35: The results of the simulations with cluster size 4 on array $\mathcal{R}$ (rectangular, 16×256 50 µm pitch pixels).

Concerning Figure 5.35b, on the other hand, it is possible to use this figure to have a glimpse on an effective particle rate capacity of this architecture. In this flavour, the efficiency is yet 99.5% at 17.5 MHz cm^{-2}, in line with the applications discussed in this thesis.

Ghost readouts

The several simulations discussed here can be used as a reference to benchmark the performances of the hash architecture. The major theoretical weakness of the proposed design are the ghost addresses, which slow down the readout and increase the power consumption: this section presents an estimation of the share of ghost readouts out of all the total readouts performed. As it was discussed before, this does not represent a major bottleneck for the maximum architecture particle rate, but anyway limits its optimal functioning.

As a reference, $\mathcal{R}$ was taken; i.e., the array with the same hash geometry that was developed in RTL code, and the ratio between the ghost readouts performed and the number of actually correct readouts was computed. This estimation was performed in the optimal region for the hash architecture, at relatively low rates, before the onset of the whole array reset[12].

The results can be seen in Figure 5.36. The fraction of ghosts grows as expected with the increase of the particle rate; nevertheless, their fraction is very small (consider that in the rightmost point reported in the plot there are 120 ghost per million correct addresses when the particle rate is 70 MHz cm^{-2}). Figure 5.36 shows that for a hit rate below 50 MHz cm^{-2} the ghost rate is lesser than 20 ppm; assuming this ghost percentage and a flow of 10 MHz cm^{-2}, the overall ghost rate is expected being lesser than 200 Hz cm^{-2}: with

[12] Actually, the reset level used here was higher than the optimal one to gain insight on this peculiar behaviour of the readout system.

good approximation (i.e., randomness in their spatial distribution) these spurious data can be considered as noisy pixels firing at random.

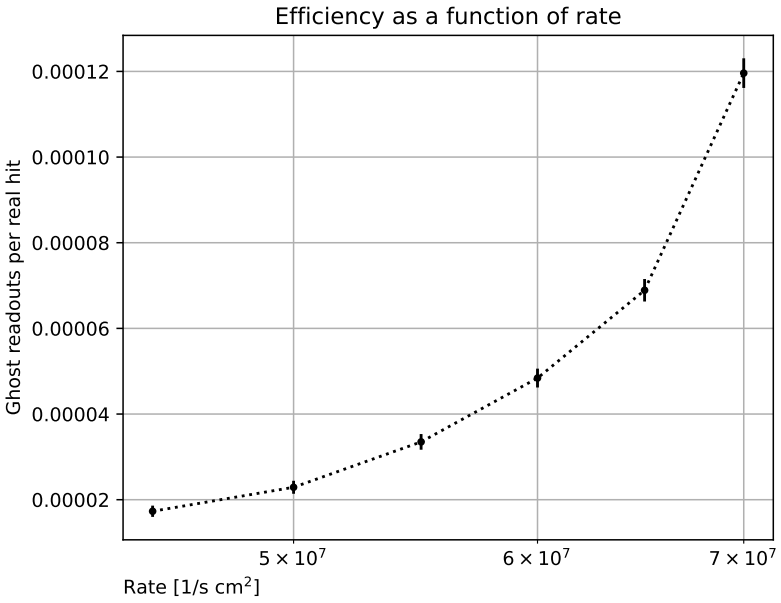


Figure 5.36: Fraction of ghost readouts out of the total ones in the $\mathcal{R}$ array.

5.3.3 RTL simulation and Timing Validation

Although using the power of C++ it is possible to simulate the behaviour of the architecture with high statistics, the actual digital architecture discussed here is developed in SystemVerilog, and only a properly tuned SystemVerilog simulation can validate the digital logic; consequently this tool has been used. First, a validation of the power calculations reported in Section 5.2.1 has been performed; then via a simulation it was verified that the architecture actually behaves as planned, without any major issue. Finally, SystemVerilog was used to validate the C++ Monte Carlo software described in Section 5.3.2.

Functionality verification

The most important simulation developed in SystemVerilog is a functionality simulation. The simulation structure is simple and much like the one presented in Section 5.3.2. Architecture is fully simulated (according to the very same compile-ready code that can be used for the physical place-and-route of the device), and then a testbench is run to analyse the architecture behaviour. Thanks to the highly cusotmizable code that was developed in order to allow a scalable design, all the matrices introduced at the beginning of Section 5.2 were simulated. Simulations include both single-hit clusters and several-hits clusters; distributed either with a uniform or gaussian-shaped distribution, and with either one only or two bits memory per pixels.

The testbench injects some data directly in the pixels, artificially raising the signal of the output of the analog front end; then data are read just before the serialization routine necessary for the device to communicate with the external world, verifying the correct functioning of the whole digital chain from the pixel to the serializer input. Delays are simulated taking into account both the wiring delay and the gate delays, hence implementing a linear approach like the one discussed in Section 5.2.1, considering only the vertical distance.

By looking at the results of this simulation, changing the parameters, it was possible to verify that:

- All the readout logic, as described in Section 5.1.5, works as expected: both the periphery and the array behave as planned during a readout routine. All the timing diagrams are respected;
- The architecture correctly complies with possible ghost hits inside the array. Ghost readouts create no actual data, and readout information is not further propagated; moreover the presence of ghost hit does not impede the subsequent readout of correct addresses;
- The readout mechanism is not impaired by hits arrival during the readout phase; as the distance between readout-and-reset is one clock cycle distance hits rarely are lost or duplicated;
- The readout mechanism is safe from the timing point of view: the periphery state machine has some dead cycles granting that the pixel selection operation terminates successfully;
- Glitches leading to undefined states or unwanted loops between statuses are not created and the architecture recoveries from errors during the readout.

The main issue in the running of such simulations is the computation time: a SystemVerilog simulation takes hours to simulate few hundred events while, on the same machine and in the same amount of time, its C++ counterpart can simulate millions of events. For this reason, this simulation was not used to actually compute quantitative statistics on device performance but as a guideline to find mistakes in the C++ modelling of the architecture or in the readout logic itself, and to verify that the architecture behaves as planned. The results of the two types of simulation, as described in the next section, agree.

C++ Verification

To verify the C++ simulation, both the RTL simulation and the C++ simulation were injected with some predefined hits, and the response was compared in terms of timing. In particular, it was possible to verify that the time is the same in both simulations (or actually the RTL simulation is faster, hence the C++ simulation is conservative) between the following *events*:

- the arrival in periphery of a hash code and the beginning of the actual readout routine when there were no hit before this one;
- the pixel sending data and resetting with respect to the beginning of readout routine in the periphery;
- the arrival of the hash code update after a pixel reset;

- two subsequent readouts in various situations (same column, same row, different column and row);
- a ghost readout in between correct readouts;
- address generation after modifications in the hash code;

Given this check, it is possible to consider the C++ simulation reliable; hence, assume that a SystemVerilog simulation performed after the synthesis and before the place-and-route behaves similarly even for a large number of injected hits.

5.4 Further improvements

The hash architecture flavour presented in this chapter is the safest and most robust one among those investigated, and it suits a first future silicon run. Nevertheless, several modifications can help boost power and timing performances: the resulting design, more error-prone than the one discussed here, suggested postponing these changes to a possible future run, after the first silicon run will be characterised and verified. In this section, some of the main improvements the hash architecture may undergo if further developed are presented: in Section 5.4.1 an improvement to the decoder performance is illustrated, in Section 5.4.2 an alternative trying to gain the best out of the hash projections is discussed, and in Section 5.4.3 an alternative pattern to perform readout-and-reset is presented.

The ideas presented in this section were sketched while creating the hash architecture but were not fully validated. Hence, this section cannot be neither exhaustive nor satisfying. It is likely that upon further investigation some of these ideas will be found to be unfeasible or not at all advantageous. All of these ideas need intensive Monte Carlo simulations (similar to the ones discussed here in Section 5.3) in order to understand them fully and evaluate their potential, their applicability, and the hidden issues that might arise from their implementation.

5.4.1 A faster decoder

As noticed in Section 5.3 the architecture bottleneck is not due to the presence of ghosts (as was initially expected), but to the speed of the hash decoder, which takes several cycles to produce a valid address to test. By implementing a faster decoder, it would be possible to move the efficiency shoulder in Figure 5.30 further to the right towards higher rates. The two implementations discussed here require higher power consumption in the periphery and more space compared to the one discussed, as well as add complexity to the overall design.

Faster clock

The easiest solution of this issue is using a faster clock for the decoder. In the current architecture implementation, for simplicity, the same clock is used throughout the whole

periphery, and a limit for the frequency of such clock was calculated by looking at the signal delays when the readouts are performed. However, in a future implementation, a faster clock could be used for the hash decoder, that is actually interfaced only with the column state machines that perform the readout, while it is completely decoupled from the array itself.

Parallelisation

Another way to improve the decoder speed relies on parallelisation: having multiple decoders working on the very same hash code at the same time looking for valid addresses. This might be done at different levels:

- One decoder can create its tentative address by looking at P_X and P_Y projections validating the address on P_U and P_V, the others might use other combinations, such as creating tentative addresses using P_X and P_U and validating using the other projections. This way up to 6 parallel decoders working at the same time can be implemented (by using four hash projections). The calculation of the time gain of this approach is not straightforward, as the events are strongly correlated and in the physical implementation discussed here a pair of projections is not optimal. Nevertheless, the addition of such parallel decoders would improve the performances of the decoding operation. The architecture would need to account for the possibility that different decoders may have a valid address at the same time;
- One decoder can work as described in Section 5.1.5, while the others might start masking a fixed number of rows. This way, the decoders would work on different rows, and the probability for at least one to find a valid address in a clock cycle becomes higher. Also in this case, the probability to find an address in a clock cycle is not easy to calculate because of strong correlations, but an arbitrary large number of decoders could be used at the same time. This approach would need a good multiplexing pattern to keep on looking for addresses considering all possible tentative addresses after a successful readout.

5.4.2 Approach alternative: the derivative approach

Another possible improvement relies on a slightly different architecture *per se*. When the present architecture tries to generate an address by looking at the hash code in the periphery, it uses only the current hash code. This rather simple approach does not use all the available information. The performances would impbe enhanced by using the difference between the current hash code and the previously sampled last one.

In order to understand how this might be useful, it is possible to consider a complicated hash code, with already several high bits, and suppose the decoder is currently looking for a valid address among the tentative ones. If in this moment, because of a new hit in the array, the hash code changes, by looking both at the current hash code and at the difference between the current hash code and the previous one, less cycles would be necessary to find a valid address and reduce the number of ghosts. This technique can be useful to stabilise the

number of cycles needed to generate a valid address, as it would improve the performance of the decoder whenever the hash population is relatively high. For this approach to be implemented, two delicate points should be taken into account:

- how to translate this algorithm in a digital circuit, with very limited space and power consumption capability;
- how to ensure that the logic is not flawed if there are multiple hits between two subsequent samplings of the hash code.

Also in this case, Monte Carlo simulations are necessary to tackle the second point and ensure the architecture stability.

5.4.3 Reset-without-readout

By considering that via Monte Carlo simulations (both system Verilog and high-statistics C++) ghost readouts were hardly spot (see Figure 5.36), as they start flooding the architecture when it is already getting clogged by the valid address research time, readout-and-reset could be further optimised. For example, instead of implementing a clustering architecture where every pixel region is first read and then reset, it would be advantageous to create a single-pixel pixel region where readout is not necessary as the periphery assumes that its valid address is correct. The advantage of this approach would be to increase the *readout* speed; i.e., reducing the time that the periphery takes to know that there is a hit somewhere in the array, from four clock cycles in the actual implementation to two clock cycles. The disadvantage would be the creation of some fake hits, as the ghost addresses would be taken as valid. This approach might be interesting in situations where fake hits are not too problematic; e.g., in hodoscopes where correlations can help selecting the actual hits and rejecting fake ones.

This idea is supported by the study on the ghost hits that was performed and is reported in Figure 5.36: if the fraction of ghost hits is this low, then the assumption they do not exist introduces a fraction of fake hits that is comparable with, if not below of, the usual number of fake hits due to the sensor electronics noise. Even without considering that an architecture that does not address pixels for the readout works twice as fast, in the array $\mathcal{R}$ the fraction of ghost hits would be below 1 every 10'000 hits at $50\,\mathrm{MHz\,cm^{-2}}$.

A more concrete possibility is to reduce the number of signals used to perform the readout: the routine used is particularly safe from a timing point of view (as discussed in Section 5.2.1) by using two vertical signals to trigger the readout and the reset of the pixel region. In the future, the possibility to have a faster approach where the data sending process and the internal reset are triggered on the edges of a single signal and the periphery samples the hitmap on its clock edge might be explored, similarly to what is currently done by ALPIDE and described in Section 4.2.2.

Lane	Implemented hash	No readout	No row selection
Hash code creation	4	4	4
Row selection	1	1	-
Read request	1	-	-
Hitmap communication	1	-	-
Sync	1	-	-
Reset request	1	1	1
Total	9	6	5

Moreover, it may be possible to remove the row selection signal for the reset, thus reducing dead time and enabling parallelisation between columns, hence improve readout speed, as well as simplify the pixels and their wiring. In this implementation, the reset would be broadcast to a full column, and also the overhead `HITMAP`, that is needed to pick ghosts from correct addresses in a standard-flavour architecture, would be lost. This might lead to missing hits, if the hash function was not decoded properly before the reset of the lane (hence, a better performing decoder would greatly improve this particular flavour of the hash architecture), and a quantification requires proper Monte Carlo simulations that were not performed yet to focus on the safest version.
These approaches, besides improving the timing performance of the architecture, would also benefit the power consumption, reducing the overall number of long lane transitions. Some algebraic computations are reported in Table 5.6, that focuses on one hit only.

Table 5.6: Long-lane transitions in the hash architecture implemented as well as the two possible bolder versions for the readout.

Chapter 6

Conclusions

This chapter summarises and gives some insights on various results contained in this work, both analytical and obtained through Monte Carlo simulations, and links to the proper chapter, section, table, or figure for the discussion on how these results have been obtained. The reference sensor is the one implemented (also called $\mathcal{R}$ through this work), with 4 pixels composing a super-pixel, a pixel pitch of 25 µm, and a height of 1.28 cm (the width is 0.8 mm but several independent sectors can be aligned on that side). The chosen technological node is LFoundry 110 nm and the all the calculations were performed on a synthesized design, therefore the figures are coherent.

First, some calculations on the area occupied by the digital readout electronics are presented, then the calculations on the power consumption are summarised, and lastly some remarks on the maximum particle flux that such an architecture might withstand are reported. In all these sections, the sensor used for comparison is the ALPIDE sensor, described briefly in Section 2.4.1, and more specifically from a digital readout point of view in Section 4.2.2. In the last subsection of this chapter, some conclusive thoughts on this thesis work are outlined taking into consideration also the work performed for ARCADIA radiation resistance experimental campaign.

Area considerations

The total area occupied by the transistors in the implemented hash architecture, where a pixel region is capable of managing four pixels, is reported in 5.2 as 500 µm^2, and in and 5.3 as 250 µm^2 for the version without configuration. This means, in comparison with the ALPIDE data reported in Figure 4.9 (about 1000 µm^2 per four pixels), that the complete version of the hash pixel that includes the configuration is half as large as the ALPIDE digital pixel; hence, smaller pixels are possible. A reduction of a factor 2 on the digital electronic area, according to algebraic calculations, can help shrink the pixel pitch of about 10% while keeping the very same dimensions for the diode area and for the analogue front end.

Power Performances

The main contribution to ALPIDE power consumption can be inferred from Figure 4.10 as the digital idling power consumption, simulated approximately 4 times higher than the analogue power consumption. For the hash architecture, the power consumption was estimated without taking into account the contribution of the periphery, and is shown in Figure 5.26a. The reduction in the number of transistors inside the pixel leads to an energy-efficient design even when idle, and the estimated value is below $10\,\mathrm{mW\,cm^{-2}}$ by assuming the use of the ALPIDE analogue front end. This performance is promising and pushes forward for future development from the place-and-route side, which would allow a direct comparison between the two.

Particle Flux

The particle flux that can be correctly disentangled by the hash architecture that was implemented can be seen in Figure 5.35b in the very pessimistic scenario where the cluster size is equal to four pixel-regions. This plot shows very good efficiency up to $\approx 2\,\mathrm{MHz\,cm^{-2}}$. According to [29], the maximum flux that ALPIDE chip is subject to is below $2\,\mathrm{MHz\,cm^{-2}}$, hence the hash architecture could be used in a tracker for an experiment like ALICE without the need for further improvement from the particle rate point of view.

Fake hits

Using the ALPIDE chip performance as a benchmark [120], it exhibits exceptionally low noise, far better than 10^{-6} pixel per readout cycle per array, in fact reaching levels of about 10^{-8} pixel per readout cycle per array, depending on the comparator threshold level. In actual use in the ALICE experiment, the ALPIDE sensor mostly operates in a 50 kHz - 200 kHz readout cycles per second range. for a reference 100 kHz readout trigger rate, a noise level of 10^{-8} pixel per array per readout, and considering that ALPIDE array is made of 5.2×10^5 pixels, the expected noisy pixel flow is therefore equal to about $100\,\mathrm{kHz}\cdot10^{-8}\,\mathrm{pixel^{-1}}\cdot 5.2\times10^5$ pixels = 520 Hz per array. As the ALPIDE pixel array covers an active area of about $4.5\,\mathrm{cm^2}$, the noisy pixel flow per unit area is on the order of $115\,\mathrm{Hz\,cm^{-2}}$.

The ALPIDE noise rate is comparable with the ghost generation rate reported in Figure 5.36. This toy exercise shows how, while in the HASH architecture a complete verification mechanism was embedded to avoid the ghost readouts, so to get a *perfect* output stream, this may not be actually necessary in practical use scenarios, permitting to further increase the speed and the power performance of the architecture, as discussed in Section 5.4.3.

Final remarks

Particle tracking plays a vital role in experiments in fundamental physics, applied physics, medical physics, and industrial applications. The scientific community is clearly interested

in silicon trackers as they are capable of achieving µm spatial resolutions with low power consumption, low material budget, and good resistance to radiation. On the other hand, the imaging industry (CMOS cameras) heavily invests in similar sensors. Among silicon sensors, the monolithic sensors (MAPS), where the particle detection and signal management are performed on the very same silicon die, are promising because of their lower price and power consumption compared to CCD and hybrid solutions. At the time of writing, one of the best performing MAPS produced so far for scientific applications, ALPIDE, is included in several scientific setups. The ARCADIA project, a call of INFN CSN5, is developing a cutting-edge multi-purpose sensor that might be applied in several experiments ranging from tracking at future lepton collider, proton computed tomography, and particle identification in experiments in outer space.

The development of such an ambitious device required an optimisation of the state-of-the-art from every point of view and it took advantage of the distributed knowledge and know-how of several INFN groups in Italy. The sensor design was optimised with the goal of reaching full depletion, improving timing performances and radiation resistance; the analogue and digital architectures were designed to keep power consumption as low as possible, create a small (25 µm) pitched array, and work with particle rates up to $100\,\mathrm{MHz\,cm^{-2}}$ by taking advantage of the sparsification hypothesis of the targeted applications. Additionally, custom hardware, firmware, and software were designed to ease sensor testing and evaluate its performance. On top of that, radiation hardness tests are still ongoing to study the performance degradation in harsh radiation environments, although correct functioning above 0 degrees after 1×10^{13} 1 MeV $\mathrm{n_{eq}/cm^2}$was measured and preliminary results agree with the simulation data obtained with TCAD.

The hash architecture, developed inside ARCADIA collaboration as an alternative readout architecture, is an innovative approach to reduce the power consumption of the digital readout architecture and the pixel pitch, maintaining a good sensor scalability: the hit positions are reconstructed by using a set of projections on different axes. To analyse the performances of this architecture, a proper mathematical formalism has been developed and used to understand how to find the best set of axes for the projections. The complete architecture was developed at a Register Transfer Level (RTL), and was hence verified by hardware-specific Monte Carlo simulations. Moreover, a faster-running C++ simulation was developed in order to create a behavioural model of the whole array and allow to estimate the rate capability. These computational tools were used to investigate slightly different implementations of the architecture, as well as different aspect ratios and pixel pitches. In order to validate the hash architecture, a complete place-and-route is necessary to estimate the timing and power performances more precisely. This level of development should be reached in the near future and have more precise estimates on the potential advantages of this architecture in comparison with the one used for the ARCADIA main demonstrators and the ALPIDE sensor.

Appendix A

Radiation Damage

As discussed through this thesis, the performance of any silicon sensor or tracker used in radiation harsh environments, like in outer space or in a High Energy Physics experimental hall, will be strongly degraded over time. Different types of radiation quanta (photons, electrons, protons, neutrons, ions) lead to different effects on silicon devices. A full discussion of radiation damage can be found elsewhere [139]–[141]. Here the most important aspects of radiation damage of semiconductor detectors are recalled, as an experimental campaign is described in Section 3.3.

As MAPS are the most recent approach to silicon detectors, intensive radiation studies do not have the massive literature that both CCD and hybrids have. Nevertheless, studies have been performed for more than two decades [142]. The fact that the electronics is embedded in the very same die where charges are produced make these kinds of detectors less radiation-resistant than the alternatives.

Radiation damage to a silicon integrated circuit is a result of two energy deposition mechanisms: ionization and bulk damage. Some damages are due to a single track lucky enough to modify the behaviour of the circuit (the **S**ingle **E**vent **E**ffect or SEE, described in Section A.1); while the amount of energy deposited over time is quantified, respectively, by **T**otal **I**onizing **D**ose (TID) (see Section A.2) and **D**isplacement **D**amage **D**ose (DDD) (see Section A.3).

A.1 Single Event Effect

One of the main advantages of digital electronics over the analogue one is the resistance to noise: whereas an analogue signal easily varies due to voltage fluctuations within the circuit, a digital one is more stable thanks to the finite distance between the high and the low level in the reference technology. Nevertheless, if a voltage swing is high enough, the effect of the noise becomes measurable and can be detrimental for the correct functioning of the circuit logic. These voltage swings are likely due to the passage of ionizing particles

that generate electron-hole pairs inside the device. These are called **S**ingle **E**vent **E**ffects (SEE) and are mostly, but not exclusively, due to the passage of low-energy ions (that have a denser ionization path). In the vast majority of cases this process has no effects, as the generated charge is comparable with the electronic noise. The actual measurable effects are grouped into two categories: soft errors and hard errors. Soft errors are those that can be recovered either with time or with a full reset of the device, hard errors, on the other hand, lead to a permanent damage on the device.

In this section, a presentation of these errors and their grouping is reported, and then a discussion on how they can corrupt the correct functioning of a MAPS employing the hash architecture discussed in this thesis is detailed. A complete description of the theory of SEE can be found elsewhere [141].

A.1.1 Soft Errors

Soft errors are grouped in two categories:

- **S**ingle **E**vent **U**psets (SEU) happen when the impinging particle modifies the charge currently stored in a memory. An example of an SEU can be seen in Figure A.1. They are the most common SEE, and their effect can be limited by using some error correction algorithms;
- **S**ingle **E**vent **T**ransients (SET) happens when the impinging particle causes a swing somewhere in the circuit voltage. After the transient, the circuit comes back to its original state. Nevertheless, this unexpected modification in the voltage is transported along the circuit, and might cause unintended behaviour somewhere, or be latched in a memory.

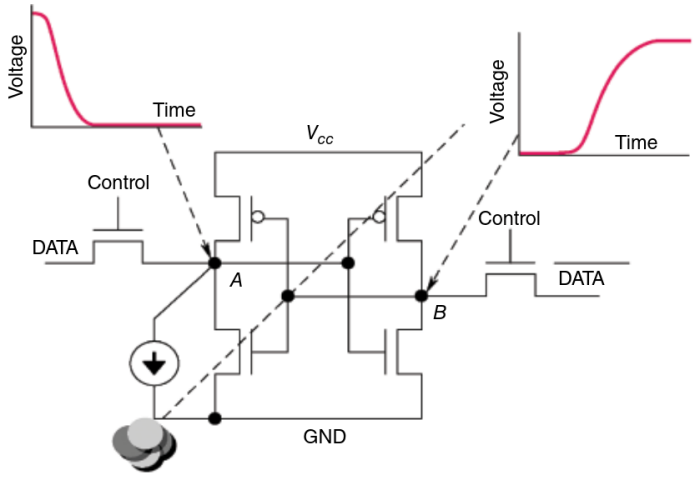


Figure A.1: An example of a SEU on a SRAM (static random-access memory) with 6 transistors. Here the generation of charge on the NMOS on the bottom left changes the value in memory from 0 to 1. The physical effect of the ionizing radiation is represented through the current generator. Taken from [143].

A.1.2 Hard errors

Hard errors are unrecoverable, and usually grouped as:

- **S**ingle **E**vent **L**atchups (SEL) that open a low-resistance current passage from the power supply to the ground in 4-layers structures (pnpn), dangerous as the high current might lead to high temperatures in the silicon regions and in the metal traces that can melt;
- **S**ingle **E**vent **B**urnout (SEB) happens whenever the produced charges start a positive feedback cycle, creating more charges. This ultimately leads to a damage in the structure of silicon, and is peculiar of a power transistor;
- Single **E**vent **S**nap-back (SES) is similar to SEL but in a single transistor, leading yet to very high currents. It can be triggered only in the presence of an external circuit providing current, and is peculiar of input and output stages;
- Single **E**vent **G**ate **R**upture (SEGR) happens when the highly ionizing impinging particle strike results in a breakdown of the gate oxide, greatly increasing the leakage current.

A.2 Total Ionizing Dose Effects

One category of radiation effects in a silicon device is called **T**otal **I**onization **D**ose (TID) Effects. In this section, some of these ionizing dose effects on a silicon sensor are summarised, as I have not participated directly to TID studies of the ARCADIA chip and test structures, that are currently ongoing. The reference work for this section is [139].

A silicon tracker is obtained by alternating several layers of silicon with various values of doping, oxides, and metals. Silicon oxide is used to isolate the conductive layers from each other, and in a sub-micron technology this oxide is present both in horizontal layers used to separate different silicon planes and vertically for other structures (like for example in the **S**hallow **T**rench **I**solation STI structures used for electrical isolation [139]). The oxide, above the other materials, is the most susceptible to damage by TID. In Section A.2.1 an example of a device cross section is reported with a discussion on why oxides are vulnerable to ionizing dose.

When an ionizing particle crosses a sensor, it releases energy, creating electron-hole pairs, that can leave behind some defects. These defects are actually produced via several different processes. The most important ones are summarised in Section A.2.2. In Section A.2.3 a brief presentation on how the defects created in a gate oxide affect the transistor function is given, and in Section A.2.4 a discussion on the effect of defects in STIs is reported.

A.2.1 Oxides in a CMOS detector

A cross section of an n-channel MOSFET in CMOS technology can be seen in Figure A.2. Whenever an ionizing particle crosses such a device, it creates charges in the silicon and in the oxide. The silicon is a semiconductor and due to the presence of an electric field, or simply via diffusion, both positive and negative charges drift towards nearby conductors. The oxide, on the other hand, acts more as an insulator: in particular the holes hardly move even in the presence of very high electric field. The mobility values for holes and electrons in native silicon and silicon oxide around room temperature are[1]:

$$\begin{aligned} \mu_e^{\mathrm{Si}} &= 1350 && \mathrm{cm/V{\cdot}s} && [45] \\ \mu_h^{\mathrm{Si}} &= 480 && \mathrm{cm/V{\cdot}s} && [45] \\ \mu_e^{\mathrm{SiO_2}} &\approx 20 && \mathrm{cm/V{\cdot}s} && [144] \\ \mu_h^{\mathrm{SiO_2}} &\approx 4 \times 10^{-9} && \mathrm{cm/V{\cdot}s} && [145] \end{aligned}$$

This means that on the one hand the charged pairs created in silicon easily drift, and the created charges get all collected by using a proper electric field; on the other hand the holes created in the oxide are easily trapped. The oxides will be the main focus for this section. In Figure A.2 two oxides are shown: the gate oxide and the STI oxides. In a sub-micron technology like the one discussed in this work, the order of magnitude of the thickness of such oxides is $O(1\,\mathrm{nm})$ for the gate oxides and $O(100\,\mathrm{nm})$ for the STIs.

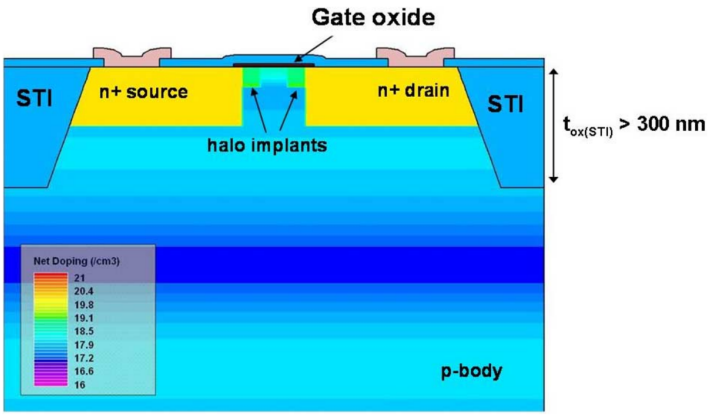


Figure A.2: A cross section of an n-channel MOSFET. Taken from [139].

A.2.2 Defects creation in the oxide

Different processes can lead to defects - localised trapped holes - in an oxide. Defects are positively charged, as electrons can move out of the oxide while the holes hardly do. These defects are grouped in literature as oxide trapped charge, border traps, and surface traps.

[1]The hole mobility in silicon oxide is quite complicated to measure; the value reported here was actually measured in thin films.

A defect falls in one of these categories depending on the distance between the trap and the closest oxide interface: oxide traps are in the oxide bulk, border traps are closer to the interface, surface traps lie on the surface. The reference distance splitting the former two categories is 3 nm, as traps closer than this can easily be filled by electrons tunneling into the traps within a time frame of the order of 1 minute [146], whereas more distant traps hardly interact directly with charges in the silicon. For this reason, surface and border traps are called switching states while oxide traps are called fixed states (see Figure A.3).

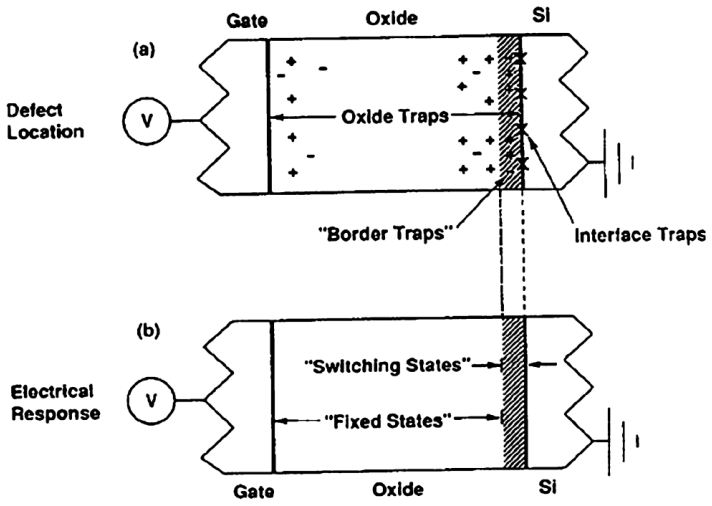


Figure A.3: Different traps in a gate oxide with their own name. Taken from [147].

The creation of a defect actually follows four steps:

- electron-hole pairs creation due to the passage of the ionizing particle;
- prompt recombination of a certain fraction of the pairs produced;
- Transport of the free carriers within the oxide;
- Formation of the traps either in a defect precursor site or via a reaction involving hydrogen.

A defect precursor is a defect in the dielectric where a hole can easily stop and get trapped, terminating its journey toward the oxide surface. This defect can either be considered a trapped charge or a border trap depending on its position. Interface traps have a more complicated taxonomy that can be found in [9]; the idea is that the moving hole interacts with a defect of the oxide containing hydrogen and this reaction releases protons, that start diffusing or moving due to the electric field reaching the oxide surface. There they interact with the H-passivated dangling bonds resulting in a defect [139].

A.2.3 Defects in gate oxide

Fixed states

Whenever there are positive traps in the body of the oxide separating the gate from the underlying silicon (fixed states), the threshold voltage needed to open the conductive channel decreases accordingly. If the transistor is an nMOS, a positive voltage is necessary to create the channel; the presence of positive charges in the gate oxide enhances the applied voltage, hence the transistor starts conducting when the applied voltage is lower than the nominal (non-irradiated) value. On a pMOS, on the other hand, a negative voltage is used to open the conductive channel, thus the trapped positive charges hinder the channel creation, leading to a more negative threshold voltage. This effect is depicted qualitatively in Figure A.4.

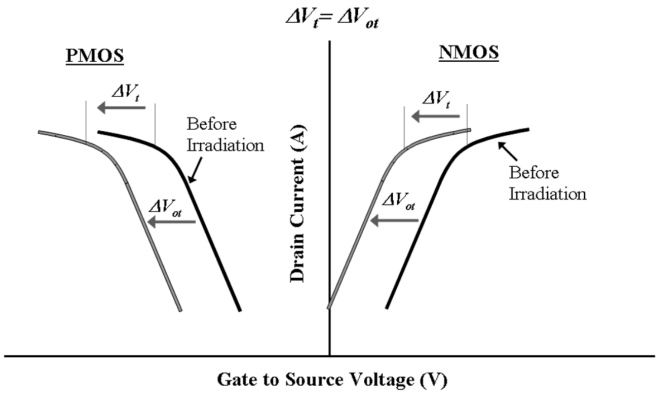
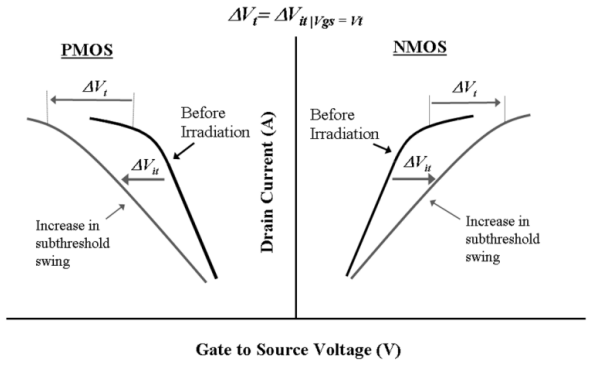


Figure A.4: The effect of positive charges on both nMOS and pMOS in terms of threshold voltage. Taken from [139].

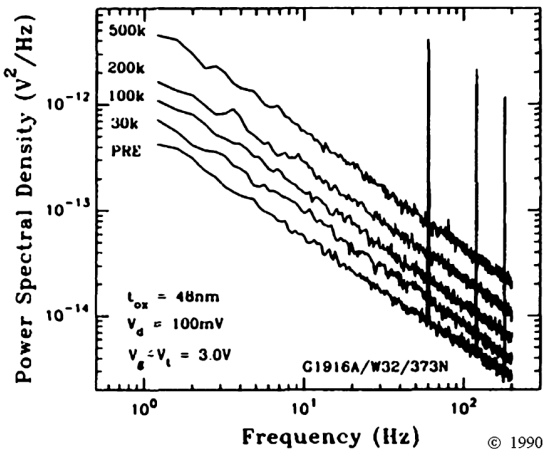
This effect, although being verified experimentally in numerous experiments [148], is not so important in sub-micron technologies. In fact, in a technology like the one discussed in this work, the gate oxide thickness is of the same order of the threshold set as a reference for the oxide traps: this means that any defect in the oxide can interact with the electrons in the silicon.

Switching states

Concerning switching states, these defects can interact with the underneath silicon layer; they can be either charged or not depending on the applied bias. Their effect on the transistor depends on their current charge. The presence of these defects have two main effects: the increase in the subthreshold swing, and the increase of the $1/f$ noise. The former effect is a restrained slope in the curve representing the current as a function of V_{GS} (see Figure A.5a); the latter can be seen by constructing the noise spectra after irradiation (see Figure A.5b). Both these effects are caused by the repeated charging and discharging of these state during transistor operations.



(a) Increase in the subthreshold swing caused by switching states in the gate oxide. Taken from [139].



(b) Increase in the $1/f$ noise caused by switching states in the gate oxide. Taken from [149].

Figure A.5: A representation of the the main effects of switching states in the gate oxide.

Deep-submicron technologies show an additional effect, called the **R**adiation **I**nduced **L**eakage **C**urrent (RILC), caused by the extremely thin oxide used to separate the gate from the body. In this condition, a defect in the oxide can be used by an electron to tunnel through the isolation layer, leading to a non-negligible leakage current. This effect is a function of the electric field in the oxide during radiation exposure and a plot of the leakage current as a function of TID can be seen in Figure A.6.

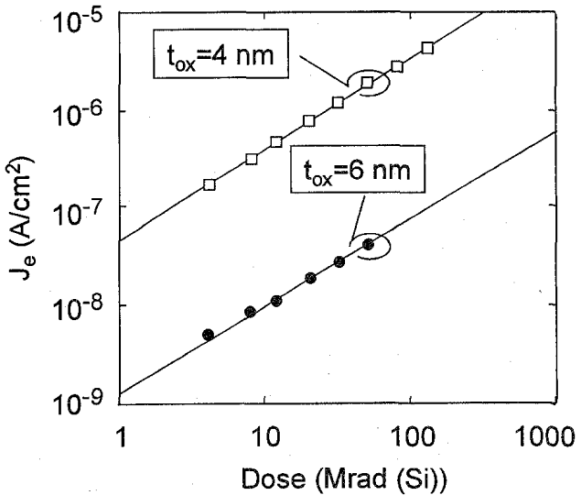
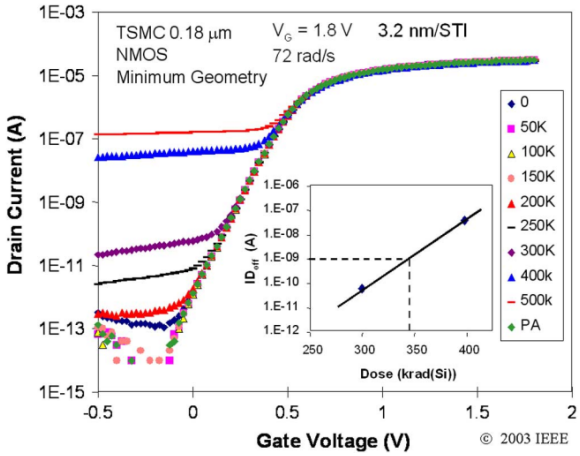


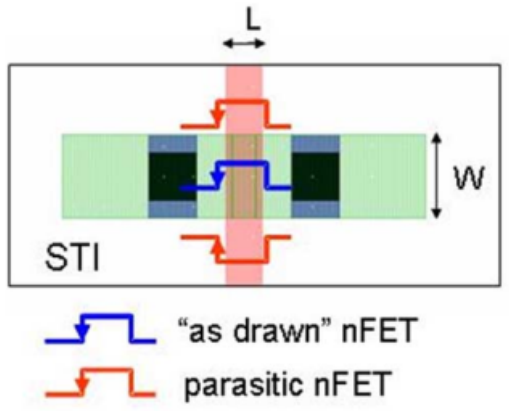
Figure A.6: The leakage current density as a function of dose for two different oxide thickness. Taken from [150].

A.2.4 Defects in STI

Experimentally, defects in STI have been shown to impact the drain-to-source leakage by creating some paths that become the dominant contributor of off-state current in n-channel MOSFET [151]. An example of such increase is shown in Figure A.7a. This effect is mostly due to the reduction in the threshold voltage and to the increase of the current flowing in the parasitic n-FET formed along the two edges of the transistor. The current, instead of flowing in the channel opened by gate voltage, flows in the adjacent oxides (see Figure A.7b).



(a) Experimental plot showing the effect of STI defects due to radiation damage in an nFET. Taken from [151].



(b) A representation of how parasitic current path are created by damaged isolation oxides, leading to a parasitic transistor. Taken from [151].

Figure A.7: The effect of irradiation of STI.

Moreover, the opening of conduction paths on isolation oxides creates inter-device leakage as well: many paths can be created between the drains of different transistors, depending on the technology used and the physical parameters. Nevertheless, inter-device leakage in literature is found smaller than intra-device leakage for several structures [152].

A.3 Displacement/Bulk Damage Effects

Whereas the oxides are the main target of total ionization damage discussed in Section A.2, non-ionising energy loss damages the silicon lattice itself. This type of damage is usually referred to as displacement (or bulk) damage, and can be caused by several particles. In this section, taken summarising the work from dr. Michael Moll [9], [153] a description of the bulk damage and how it impacts a device like the one discussed in this work is given.

In Section A.3.1 more details are given on how non-ionizing energy loss can damage a silicon lattice, in Section A.3.2 the effects of these defects on a sensor is discussed, and in Section A.3.3 an analysis on how the defects and their effect changes with respect to some physical quantities such as the temperature and the annealing cycles performed is reported.

A.3.1 Modelling

Although a silicon sensor can detect an impinging particle by counting the electrons and holes produced by its passage, this is not the only way a particle can transfer energy to the tracker medium. It may also undergo a **Non-ionizing energy loss**; i.e., the transferal of energy to an atomic nucleus dislodging it from its position in the crystal. The interaction mechanism for this to happen depends on the impinging particle type; e.g., if the particle is charged, the interaction occurs via the coulomb force, and if it is fast enough to overcome the coulomb barrier, strong nuclear interaction channels open up too; if the particle is a neutron, only the strong nuclear interaction will play out. High energy gamma rays can interact with atomic nuclei too, resulting not only in nuclear recoil, but at very high energies, also in the ejection of particles in photo-disintegration, even nuclear fission (photo-fission).

The structural damage to the lattice due to irradiation has been extensively studied over the last decades and is one of the major concerns for future experimental high energy physics (for example for the upcoming **h**adron-**h**adron **F**uture **C**ircular **C**ollider - hh-FCC [154]).

Non-ionizing damage

If an impinging particle interacts with a atomic nucleus of the silicon lattice, it may knock the nucleus out of its position only if the energy transfer is higher than the displacement threshold of about 25 eV. If this is the case, the displacement of the atom actually creates two different defects that can interact: the displaced atom (called **P**rimary **K**nock-out **A**tom - PKA) and the defect in the crystal structure left behind, named vacancy. These two defects are commonly referred to as a Frenkel pair. If the energy transferred to the PKA is high enough it may in turn displace other atoms in the lattice, creating a bowling-like cascade of displacements and vacancies referred to as a damage cluster. The moving PKA and the secondary displaced atoms may also ionize the silicon along their path. It should also be noted that if the impinging particle interacts via the strong nuclear force, the nucleus of the PKA could actually breakup into heavily ionizing fragments. The effect of these all these secondary ionisations, from simple recoils to nuclear fragments, is however transient because of the high mobility of electrons and holes in silicon (see Section A.2)A simulation of a PKA with high energy creating several clusters can be seen in Figure A.8. Defects become denser at the end of any recoil range: this is due to the fact that when recoil is slow non-ionizing reactions are favoured.

Using a non-relativistic approach, the maximum energy that can be imparted to a stationary silicon atom in an elastic collision with a particle with mass m and energy E to a recoil atom can be calculated as:

$$E_{\max} = 4E\frac{m \cdot m_{\mathrm{Si}}}{(m + m_{\mathrm{Si}})^2}$$

Considering that in silicon the displacement threshold to create a Frenkel pair is about 25 eV and that the cluster creation threshold is about 5 keV [155], a proton/neutron needs at least

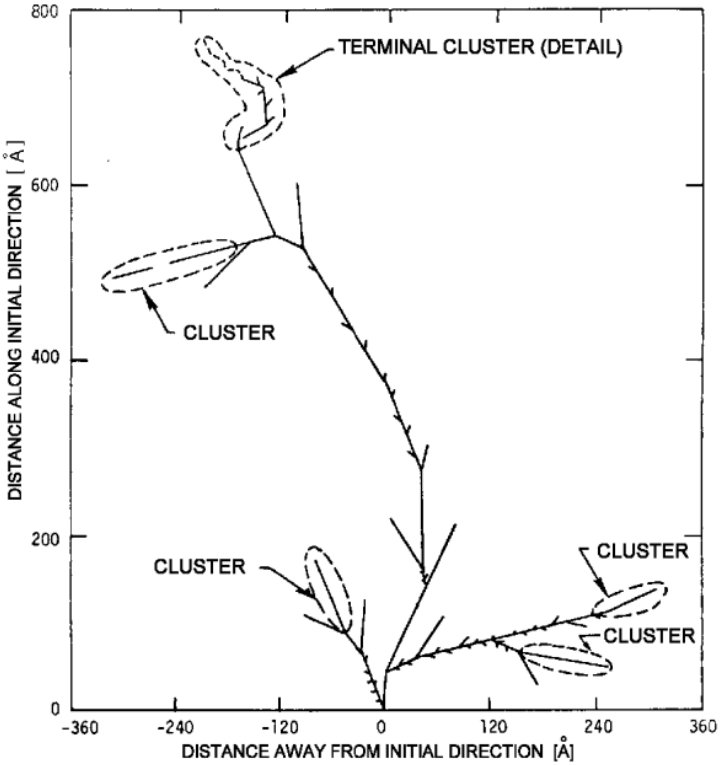


Figure A.8: The simulation of a primary Knock-out Atom with an energy of 50 keV, creating several clusters. The typical cluster diameter is about 50 Å. Taken from [155] via [9].

≈ 185 eV to knock an atom out from the its lattice position and at least ≈ 35 keV to produce a cluster.

Different particles and the NIEL hypothesis

Whereas the detection of a particle requires the transfer of energy to the electrons of the detector medium with consequent ionization or excitation, to damage the silicon lattice the particle must transfer non-ionizing energy. All particles may damage the lattice however ionizing energy loss is favoured when the particle is charged. The only common particle that is considered as incapable of creating cluster damage in silicon is the photon; 1 MeV photons from ^{60}Co do create isolated Frenkel pairs in silicon, but no clusters.

To quantify the damage caused by incoming radiation, various quantities are defined. First, KERMA is defined as the **K**inetic **E**nergy **R**elease in **MA**tter: an energy dependant quantity representing the energy that particles have deposited as kinetic energy in the lattice; i.e., potential displacement[2] [156]. KERMA can be calculated either using the weight w of

[2]The word potential is used as many Frenkel pairs annihilate just after their production.

the irradiated device as:

$$\text{KERMA}[\text{MeV}] = \Phi[\text{cm}^{-2}] \cdot w[\text{gram}] \cdot \text{NIEL}[\text{MeV cm}^2\,\text{g}^{-1}]$$

or using the number of atoms in the device:

$$\text{KERMA}[\text{MeV}] = \Phi[\text{cm}^{-2}] \cdot N_{\text{atoms}} \cdot 10^{-27}[\text{cm mb}^{-2}] \cdot \text{D}[\text{MeV mb}]$$

The above equations define and consequently relate the so-called Damage Function $D(E)$ and the NIEL(E):

$$D(E) = \frac{W_A}{N_A}\text{NIEL}(E)$$

where W_A is the atomic weight and N_A is Avogadro's number. For silicon the conversion factor is:

$$100\,\text{MeV mb} = 2.144\,\text{keV cm}^2\,\text{g}^{-1} \tag{A.1}$$

In the NIEL hypothesis, the bulk damage due to irradiation depends only on the KERMA. This means that damages can be fairly compared and scaled: for instance the damage due to high energy neutrons that make large clusters with the damage due to low energy neutrons that displace just one atom, or with the damage of a charged pion that transfers energy both via ionizing and non-ionizing processes. To make a scaling it is necessary to know the particle energy E and the corresponding NIEL or, equivalently, its damage factor D.

In Figure A.9 an experimental comparison of the NIEL and the damaging factors of several particles is shown. This equivalence greatly helps the modelling of bulk damage: it is possible to estimate the damage in a harsh environment like in hh-FCC by combining the effects of the great many diverse particles with various energy spectra that are produced by in the collision of energetic hadrons.

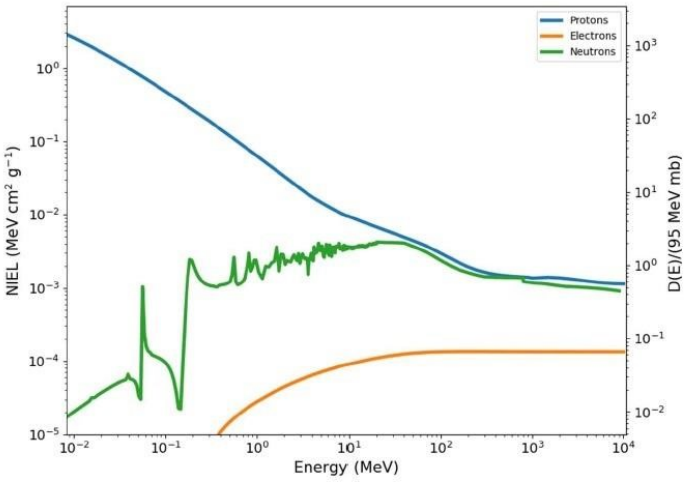


Figure A.9: Plot of the NIEL [$\text{MeV cm}^2\,\text{g}^{-1}$] as a function of energy (primary y-axis) and of the 1-MeV neutron normalised displacement damage cross-section $D(E)$ (secondary y-axis) for neutron, protons, and electrons. Adapted from [9].

The bulk damage scales linearly with the particle fluence. As NIEL hypothesis is commonly accepted, the fluence is often normalised and reported in units of 1 MeV n_{eq}/cm^2. To perform this normalisation, the hardness factor K is used: it is defined as the ratio between the damaging factor of the particle currently used for irradiation and the 1 MeV neutron damaging factor $D_N = 95\,\text{MeV mb}$. As an example, in 2023 the ARCADIA demonstrator will undergo a bulk damage experimental campaign at the SIRAD irradiation line, using 27 MeV protons for bulk damage, and the reference hardness factor is $K_p \approx 3$ [113]. The NIEL hypothesis remains an approximation and it can fail in some cases [153]. In Section A.3.3 some more details on this subject are discussed.

A.3.2 Effects on silicon

The damage on the silicon lattice produces measurable effects: as the damage increases, the performance of the device worsens. In this section three main effects of a damaged lattice are summarised: the increase of the leakage current, the modification of the effective space charge, and the charge trapping.

Leakage current

When a silicon sensor is operated, a leakage current, i.e., a current on the line used for the device biasing, is always measured. This current comes from the detector bulk but also from its *surface*; i.e., the interface between the silicon and the oxide used for isolation. Bulk leakage current has two different origins:

- The junction is depleted from the majority carriers, however minority carriers are still present and, traversing the junction, will cause leakage current. This contribution is roughly proportional to the area of the junction and is it rarely an important source of leakage current;
- Inside the depletion region, electron-hole pairs are continuously created by simple thermal generation mechanisms. This effect depends strongly on the temperature.

With what concerns the surface generated current, there are usually very high voltage gradients, hence even minor contaminations and defects on the surface can produce significant leakage currents. To avoid these, it is necessary to use some physical blockages (like guard rings) to reduce the possible current paths.

When the bulk of a silicon detector is damaged by radiation, states are created between the conduction and the valence bands. These states can then be used by electrons to jump between the bands, and this leads to an increase in the leakage current. Higher leakage current brings both an increase in noise and in power consumption. These effects can be mitigated by cooling, and some details are provided in Section A.3.3. In Figure A.10 some measurements on the leakage current after irradiation and annealing is shown for various silicon detectors changing process technology, resistivity, and conduction type. The only important parameter is the neutron equivalent fluence, thus confirming the NIEL model

approach. Usually, leakage current due to irradiation is parameterised by the current-related damage factor α defined as:

$$\alpha = \frac{\Delta I}{V \phi_{\text{eq}}} \tag{A.2}$$

representing the slope of the line in Figure A.10.

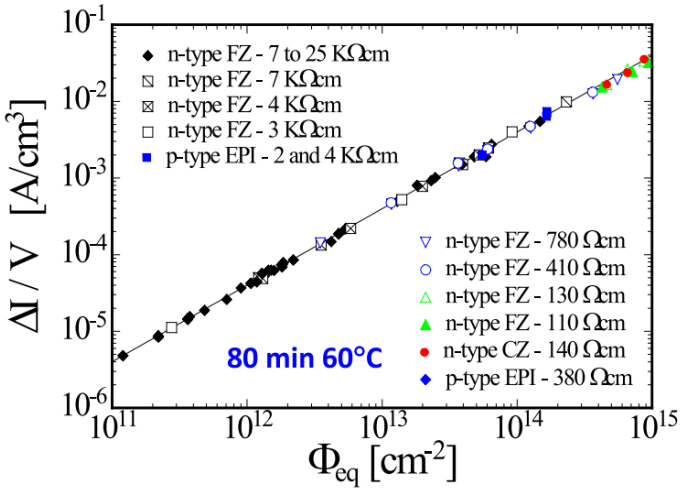


Figure A.10: Leakage current measurement with different detectors changing process technology, resistivity, and conduction type. Taken from [9] via [153].

Effective space charge

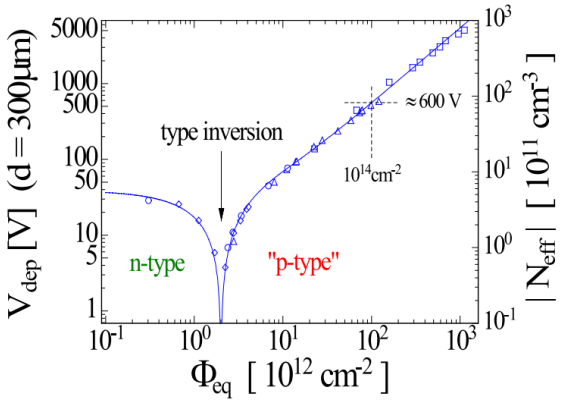
As previously discussed in Section 2.3, in a silicon tracker the internal electric field is designed using complex simulations and obtained by tuning the physical properties of the silicon used. The new states created by the impinging radiation modify the actual effective space charge and the electric field distribution. Assuming that the space charge is homogeneous (this is not always the case), the changes in the effective space charge leads to a change in the depletion voltage, that can be calculated, using Poisson's law, as:

$$V_{\text{dep}} = \frac{q \left| N_{\text{eff}} \right| d^2}{2\varepsilon}$$

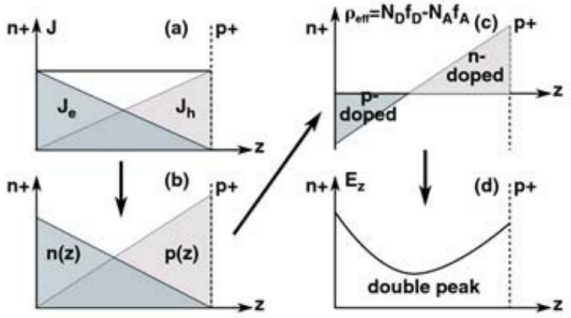
where the effective space charge is N_{eff}, d is the device thickness and ε is the silicon permittivity. The depletion voltage is a quantity that can easily be measured by C-V measurements[3]. Figure A.11a shows the variation of the depletion voltage as a function of the particle fluence [153], starting from a high-resistivity n-type base material. Irradiation causes the formation of negative space charge, that compensates the initial positive space charge, reaching very low net space charge values (this point is called **S**pace **C**harge **S**ign **I**nversion, SCSI); after this point the depletion voltage starts growing once again. This effect is peculiar for n-type

[3]The ARCADIA depletion voltage measurements on unirradiated devices were planned by simulation [97] and then experimentally measured [98].

detectors; there is no SCSI in p-type ones. If the effective space charge is too high, it will be hard to reach the depletion voltage, which value gets closer and closer to the breakdown voltage. In this case, to avoid breakdown, the device must be operated in an underdepleted mode and the detected signal is lower than what it could be.



(a) Depletion voltage as a function of the particle fluence. Space Charge Sign Inversion point is clear. Taken from [157] via [153].



(b) A representation of the double junction effect. a) current density due to leakage current; b) carrier density distribution with higher hole concentration due to lower hole mobility; c) distribution of space charge; d) distribution of electric field. Taken from [153].

Figure A.11: Two representations of effective space charge effects.

Moreover, if the bulk damage is inhomogeneous, the effective charges and relative shifts in the electric field inside the sensor volume will consequently be inhomogeneous too, making the sensor response depend on the impinging point of the particle. Another effect due to inhomogeneous bulk damage is the so-called double junction effect: some regions might have passed the SCSI while other regions are still n-type. In this case, the electric field shows a clear double peak (see Figure A.11b).

Trapping

When a heavily-irradiated device is used, the charges produced by any new ionization might get trapped in the levels created in the energy gap by the lattice defects. If the characteristic decay time of these traps is lower or comparable with the device collection time this effect is negligible but, on the other hand, if charges are not released fast enough, the signal collected is lower than the pristine undamaged one. This effect is quantified in term of a decrease of the **C**harge **C**ollection **E**fficiency (CCE), i.e., the height of the analog output of the sensor with respect to the total charges produced. This quantity is a function of trapping time, that is the time that carriers spend inside a trap. Experimentally, previous works [153] show that the inverse effective trapping time grows linearly with the particle fluence (see Figure A.12):

$$1/\tau_{\text{eff}} = 1/\tau_0 + \beta\phi_{\text{eq}}$$

and the constant β has measured values of about 4-6 $\times 10^{-16}\text{cm}^2\,\text{ns}^{-1}$ for electrons and 5-8 $\times 10^{-16}\text{cm}^2\,\text{ns}^{-1}$ for holes [153].

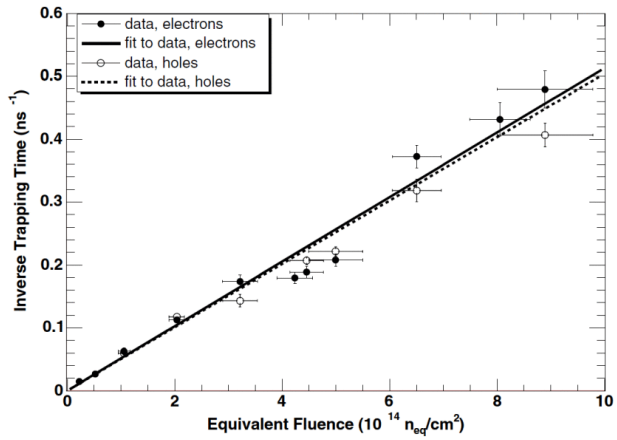


Figure A.12: Trapping time as a function of particle fluence (irradiations done with protons, but normalised to neutron equivalent). Here, measured after annealing. Taken from [158].

A.3.3 Dependencies

Most of the effects discussed in this section depend on two physical quantities: the working **temperature** of the device, and the thermal **annealing** cycles done to the device between the irradiation and the measurements.

Temperature

Temperature plays an important role for the performance of a detector, even in the absence of bulk damage. One of the paramount contributions to leakage current is the thermal excitation of electrons. If the temperature is reduced, the number of electrons that jump from the valence band to the conduction one is reduced as well. In a damaged device, temperature can play a major role in the correct detector functioning; in Section 3.3 it is described how, experimentally, the functionality of a detector is restored by lowering its temperature.

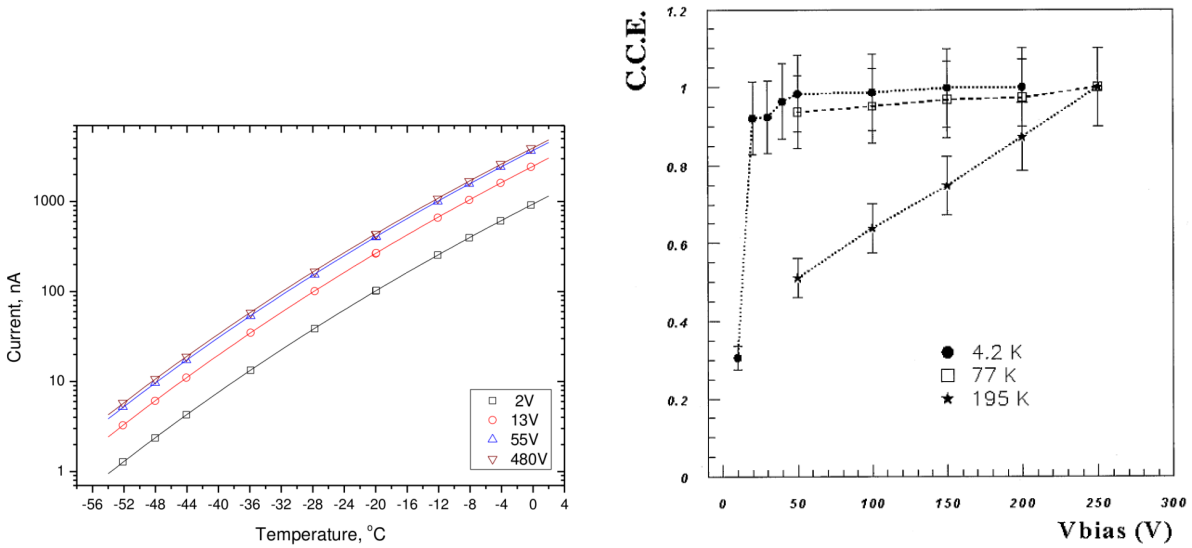
Concerning leakage current, the dependence on temperature is exponential. Theoretical calculations [159] lead to the equation:

$$I(T) \propto T^2 \exp\left(-\frac{E_{\text{eff}}}{2kT}\right) \tag{A.3}$$

This relation has been verified experimentally (Figure A.13a) and is often used to calculate the parameter named effective energy E_{eff}, that depends on the position of the mid-gap level created by the non-ionizing energy deposited.

Concerning the effective space charge, some measurements have been performed on the depletion voltage as a function of the temperature. No statistically significant dependencies were found at temperatures ranging from $-24\,^{\circ}\text{C}$ to $12\,^{\circ}\text{C}$ [160].

The trapping effect can be measured via the charge collection efficiency. Experimentally [161] it was shown that by reducing the temperature it is possible to increase the charge collection efficiency; data are shown in Figure A.13b.



(a) Leakage current as a function of temperature. Here, four different voltages used for bias. The exponential relationship can be seen. Taken from [159].

(b) Experimental measurements on the dependence of the charge collection efficiency (CCE) on temperature. Here a 350 µm silicon detector irradiated to a fluence of 1.19×10^{14} 1 MeV n_{eq}/cm^2. Taken from [161].

Figure A.13: Temperature dependence on some effects of bulk damage.

Annealing

Annealing is the process of recovering damage of the silicon lattice. It is performed by heating the device, keeping it hot for a fixed period of time. There actually are three main mechanisms leading to the annealing of defects:

- Migration: when the temperature is high enough, a defect becomes mobile. Annealing happens when a defect reaches and recombines with a counterpart (for example the PKA can recombine with the vacancy);
- Complex formation: a moving defect can combine with a different defect, different from its counterpart (else it would annihilate), and the two defects create a single complex one;
- Complex dissociation: a complex defect can vibrate and, at high temperatures, it can breakup into its simpler parts.

Each of these processes is characterised by an activation energy; a schematic representation of these three processes can be seen in Figure A.14

By looking at all the possible defects studied in silicon, each one of them has a characteristic annealing temperature, as is shown in Figure A.15. Going back to the measurable

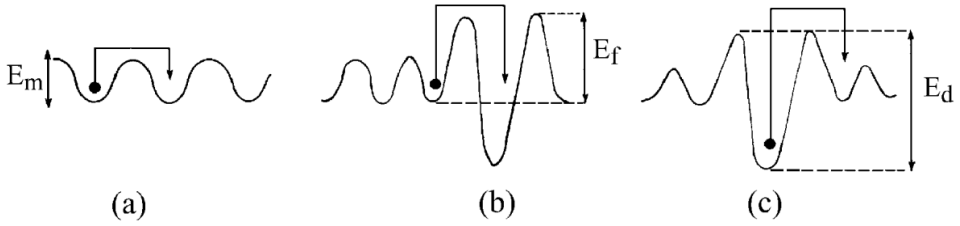


Figure A.14: A schematic representation of defect annealing: a) migration; b) complex formation; c) complex dissociation. Taken from [162] via [9].

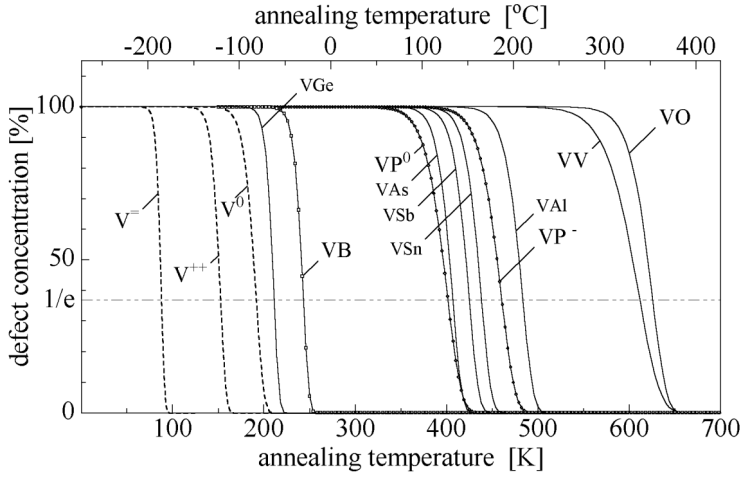


Figure A.15: The theoretical prediction of defect concentration as a function of annealing temperature, shown for different types of defect. Here an annealing time of 20 minutes was taken as a reference. Taken from [9].

effects of the bulk damage of a silicon detector, annealing has a major effect on the reduction of the leakage current. A parameterisation of the leakage current α, proposed in previous works ([9] via [153]) is the following:

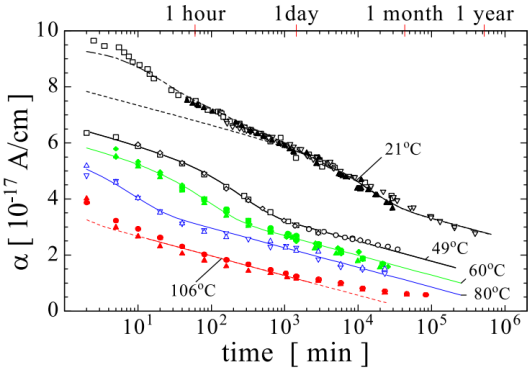
$$\alpha = \alpha_1 \cdot \exp(-t/\tau_1) + \alpha_0 - \alpha_2 \cdot \ln(t/t_0)$$

This function is in good agreement with the experimental data, as can be seen in Figure A.16a.

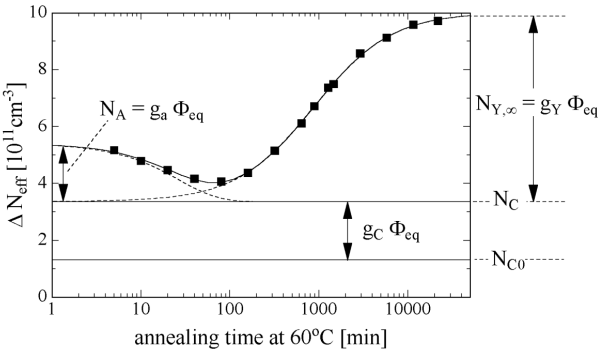
For what concerns the effective charge space (and thus the depletion voltage), the effective doping concentration changes with time and high temperatures accelerate this process. A parameterisation of the changing in the effective doping is:

$$\Delta N_{\text{eff}}(t) = N_A(t) + N_C + N_Y(t)$$

where N_C is the stable damage (not time-dependant), N_A is the short term annealing component, and N_Y is the reverse annealing component. To understand the reason behind the presence of these terms, a full discussion of the physics behind bulk damage is necessary. The time-dependant components are all exponential with time with their own timing constants, and whereas N_A grows with time (showing a reduction of the defects) N_Y is opposite.



(a) Experimental data for the running of the leakage current parameter as a function of annealing temperature and time. Taken from [163] via [153].



(b) Experimental measurements on the dependence of the effective doping as a function of annealing time at 60 °C. These measurements were performed on a 25 kΩ cm device irradiated with a fluence of 1.4×10^{13} 1 MeV n_{eq}/cm^2. Taken from [9].

Figure A.16: Annealing effect on silicon bulk damage effects.

N_Y has the longest time constant, of the order of 500 days at room temperature [9]. In Figure A.16b the effective doping concentration is shown as a function of annealing time at 60 °C. Here all the three components can be clearly recognised.

Regarding charge collection efficiency, also in this case there are two opposing effects: with annealing the trapping probability decreases for electrons and increases for holes. This means that, for some detectors, annealing might be good as the main source of signal are the electrons. Some measurements can be seen in Figure A.17.

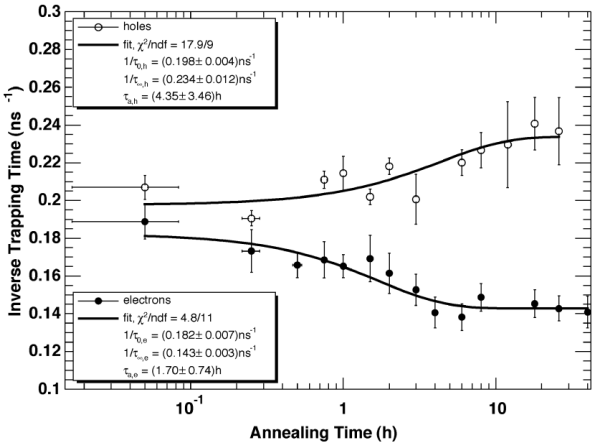


Figure A.17: Some measurements on the effect of annealing on Charge Collection Efficiency. Here a 297 µm device irradiated with 4.46×10^{14} 1 MeV $n_{eq}/cm^2$and successively annealed at 60 °C. Taken from [164].

Beyond NIEL: defect engineering

The new challenges in high particle physics need to push the radiation tolerance of silicon detectors by some orders of magnitude. For example, the long term damage of silicon sensors is to evaluated, for the next big hadron collider (hh-FCC), up to a 1-MeV neutron equivalent fluence value of $6 \times 10^{17}\,\mathrm{cm}^{-2}$ [154]. To reach this, the community is trying to go beyond the NIEL model, by taking into account the fact that each damaging particle actually has a different effect.

This is the idea behind **defect engineering**, where the impurity content is engineered in an attempt to control the damage effects of certain types of particles. This method has been used by the RD48 collaboration [165] using various methods for silicon growth and different inclusions in the crystal. The results they obtained strongly depend on the particle used for the irradiation. As an example of defect engineering, by adding oxygen to the crystal clear advantages were measured in the cases of proton- or pion-induced damage while no benefits were measured when irradiating with neutrons. These results can be seen in Figure A.18.

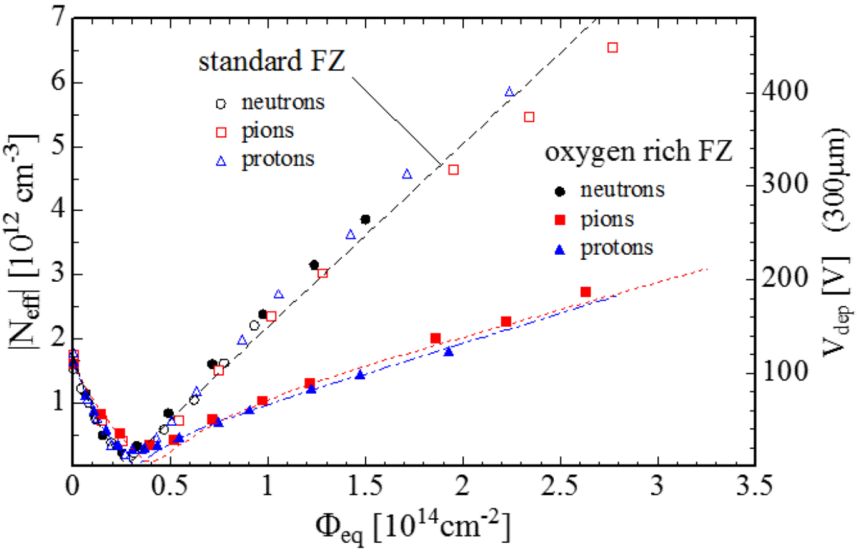


Figure A.18: The effective dopant concentration in standard and oxygen-enriched FZ silicon irradiated with different particles. Taken from [165] via [153].

Appendix B

ARCADIA-MD1 coincidences seeker

The coincidence seeker is a general algorithm developed to find events (sets of data within the same timing window) when these data come from different devices that are correctly synchronised. The implementation discussed here is developed for a hodoscope of three ARCADIA-MD1 sensors. This algorithm analyses the three separated sets of data (collected up to the moment when the algorithm is executed), one for each detector, and finds the sets of data for which there is a hit on at least two devices in a timing window. Here, some pseudo-code is presented: `data_i` represents the vector containing data, `id_i` represents an id scrolling over data, and `Twindow` is the timing window for which the hits are considered as belonging to the same event.

```
# While there still are data.
while id0 < len(data0) and
      id1 < len(data1) and
      id2 < len(data2):

  # Get the time of current data.
  t0 = data0[id0].t
  t1 = data1[id1].t
  t2 = data2[id2].t

  # If two timestamps at are inside the window
  if abs(t0 - t1) < Twindow or
     abs(t0 - t2) < Twindow or
     abs(t2 - t1) < Twindow:
       # Create the new event.
       CreateSyncEvent( start = min(t0, t1, t2) )

  else:
     # Increase index with lowest time.
     if min(t0, t1, t2) == t0: id0 += 1
     elif min(t0, t1, t2) == t1: id1 += 1
     elif min(t0, t1, t2) == t2: id2 += 1
```

The `CreateSyncHit` function is quite simple, as it just saves all the data inside the timing window. A pseudo-code representation is:

```
def CreateSyncHit( start ):
    # Create the new Event.
    newSyncEvent

    # Loop over all three data sets.
    for i in 1, ..., 3:

        # While there still are data.
        while id_i < len(data_i):

            # If inside the timing window
            if abs(data_i[id_i].t - start) < Twindow:
                # Add hit to event, go on with id.
                newSyncHit.append( data_i[id_i] )
                id_i += 1

    return newSyncEvent
```

The advantage of this algorithm lies in the fact that the index is saved across multiple runs, eliminating the need to search through old data when running the algorithm on new data. Furthermore, it allows for the identification of events with data collected between interleaved readouts.

Appendix C

A single hit through the Monte Carlo

To understand better how the simulation works, an example of the C++ simulation taking into account the time delays has been run in `debug mode` with only one event. The output of the simulation is:

```
|      Event popped     |
|      T [ns]   | X  | Y  | EVENT
|-------------|---|---|------
|            12|   0|   0|clockTick
|            24|   0|   0|clockTick
|            36|   0|   0|clockTick
|            48|   0|   0|clockTick
|            60|   0|   0|clockTick
|            72|   0|   0|clockTick
|            84|   0|   0|clockTick
|            96|   0|   0|clockTick
|            97|   7|  32|addHit
|            97|   7|  32|hashSend
|           101|   7|  32|hashArrive
|           108|   0|   0|clockTick
|           120|   7|  32|readSend
|           120|   0|   0|clockTick
|           121|   7|  32|readArrive
|           121|   7|  32|dataSend
|           122|   7|  32|dataArrive
|           132|   7|  32|resetSend
|           132|   0|   0|clockTick
|           144|   0|   0|clockTick
|           145|   7|  32|resetArrive
|           145|   7|  32|hashSend
|           149|   7|  32|hashArrive
|           156|   0|   0|clockTick

******** Printing array scoreboard: ********
```

```
******** Hits:     1
******** Pileup:   0
******** Lost:     0
******** Ghosts:   0
******** Read:     1
*****************************************
```

The color encoding is similar to the one in Figure 5.28. In purple, the only event including random number generation: the particle arrival. In red, events delayed due to signal travel time. In blue, event triggered on the clock tick. Green color is added, representing here prompt event generated by the previous one. The simulation can end on different conditions: in this case the condition given was to stop when there are no more particles to simulate and the array is completely empty (and this condition is the one used the most). The final scoreboard reports:

- Total number of hits generated;
- Number of hits that were generated in an already populated pixel (pileup hits);
- Number of hits not read but reset (this happens in the case of full array reset) (lost hits);
- Number of ghost readouts. These are readouts that did not produce an actual data to transmit and are not fake hits as they are not transmitted to the user;
- Number of correctly read hits.

Bibliography

[1] The ALICE Collaboration, K. Aamodt, and A. A. Quintana, "The alice experiment at the cern lhc," *Journal of Instrumentation*, vol. 3, no. 08, S08002, Aug. 2008. DOI: 10.1088/1748-0221/3/08/S08002. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/3/08/S08002.

[2] The CMS Collaboration, S. Chatrchyan, G. Hmayakyan, and V. K. and, "The cms experiment at the cern lhc," *Journal of Instrumentation*, vol. 3, no. 08, S08004, Aug. 2008. DOI: 10.1088/1748-0221/3/08/S08004. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/3/08/S08004.

[3] The ATLAS Collaboration, G. Aad, E. Abat, J. Abdallah, A. A. Abdelalim, and A. Abdesselam, "The atlas experiment at the cern large hadron collider," *Journal of Instrumentation*, vol. 3, no. 08, S08003, Aug. 2008. DOI: 10.1088/1748-0221/3/08/S08003. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/3/08/S08003.

[4] The LHCb Collaboration, A. A. A. Jr, L. M. A. Filho, and A. F. Barbosa, "The lhcb detector at the lhc," *Journal of Instrumentation*, vol. 3, no. 08, S08005, Aug. 2008. DOI: 10.1088/1748-0221/3/08/S08005. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/3/08/S08005.

[5] W. Atwood, A. A. Abdo, M. Ackermann, *et al.*, "The large area telescope on the fermi gamma-ray space telescope mission," *The Astrophysical Journal*, vol. 697, no. 2, p. 1071, 2009.

[6] J. Taylor, P. Allport, G. Casse, *et al.*, "Proton tracking for medical imaging and dosimetry," vol. 10, no. 02, p. C02015, Feb. 2015. DOI: 10.1088/1748-0221/10/02/C02015. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/10/02/C02015.

[7] J. D. Kurfess, E. I. Novikova, B. F. Phlips, and E. A. Wulf, "Compton imager using room temperature silicon detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 579, no. 1, pp. 367–370, 2007, Proceedings of the 11th Symposium on Radiation Measurements and Applications, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2007.04.077. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900207006493.

[8] W. Dulinski, J.-D. Berst, A. Besson, *et al.*, "Cmos monolithic active pixel sensors for minimum ionizing particle tracking using non-epitaxial silicon substrate," *IEEE Transactions on Nuclear Science*, vol. 51, no. 4, pp. 1613–1617, 2004. DOI: 10.1109/TNS.2004.832947.

[9] M. Moll, "Radiation damage in silicon particle detectors: Microscopic defects and macroscopic properties," Ph.D. dissertation, Hamburg U., 1999.

[10] M. Mager, "Alpide, the monolithic active pixel sensor for the alice its upgrade," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 824, pp. 434–438, 2016, Frontier Detectors for Frontier Physics: Proceedings of the 13th Pisa Meeting on Advanced Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2015.09.057. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900215011122.

[11] F. Glessgen, "Characterization of irradiated passive cmos sensors for tracking in hep experiments," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1046, p. 167 694, 2023, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.167694. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016890022200986X.

[12] K. Moustakas, M. Barbero, I. Berdalovic, *et al.*, "Cmos monolithic pixel sensors based on the column-drain architecture for the hl-lhc upgrade," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 936, pp. 604–607, 2019, Frontier Detectors for Frontier Physics: 14th Pisa Meeting on Advanced Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2018.09.100. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900218312531.

[13] Y. Li, "Maps for the upstream tracker in lhcb upgrade ii," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1032, p. 166 629, 2022, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.166629. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900222002017.

[14] E. Ricci, "First use of monolithic active pixel sensors for tracking particles in space," *44th COSPAR Scientific Assembly. Held 16-24 July*, vol. 44, p. 3057, 2022.

[15] M. Battaglia, C. Da Viá, D. Bortoletto, *et al.*, "R&d paths of pixel detectors for vertex tracking and radiation imaging," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 716, pp. 29–45, 2013, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2013.03.040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900213003458.

[16] F. Hugging, "The atlas pixel detector," *IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 1732–1736, 2006. DOI: 10.1109/TNS.2006.871506.

[17] L. Cremaldi, "Cms pixel detector—overview," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 511, no. 1, pp. 64–67, 2003, Proceedings of the 11th International Workshop on Vertex Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)01752-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900203017522.

[18] M. Munker, "Test beam and simulation studies on high resistivity cmos pixel sensors," Ph.D. dissertation, Bonn U., 2018.

[19] I. Perić, M. Prathapan, H. Augustin, *et al.*, "A high-voltage pixel sensor for the atlas upgrade," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 924, pp. 99–103, 2019, 11th International Hiroshima Symposium on Development and Application of Semiconductor Tracking Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2018.06.060. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900218307903.

[20] S. Orfanelli, "The phase 2 upgrade of the cms inner tracker," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 980, p. 164 396, 2020, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2020.164396. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900220307932.

[21] A. Yüncü, *A truly cylindrical inner tracker for alice*, 2022. arXiv: 2212.03165 [physics.ins-det].

[22] A. Starodumov, P. Berger, and M. Meinhard, "High rate capability and radiation tolerance of the proc600 readout chip for the cms pixel detector," *Journal of Instrumentation*, vol. 12, no. 01, p. C01078, 2017.

[23] S. Mattiazzo, F. Baruffaldi, D. Bisello, *et al.*, "Impact: An innovative tracker and calorimeter for proton computed tomography," *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 2, no. 4, pp. 345–352, 2018. DOI: 10.1109/TRPMS.2018.2825499.

[24] J. Bregeon, "Design and performance of the silicon strip tracker of the fermi large area telescope," *Journal of Instrumentation*, vol. 6, no. 12, p. C12043, Dec. 2011. DOI: 10.1088/1748-0221/6/12/C12043. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/6/12/C12043.

[25] T. Corradino, G.-F. Dalla Betta, L. De Cilladi, C. Neubüser, and L. Pancheri, "Design and characterization of backside termination structures for thick fully-depleted maps," *Sensors*, vol. 21, no. 11, 2021, ISSN: 1424-8220. DOI: 10.3390/s21113809. [Online]. Available: https://www.mdpi.com/1424-8220/21/11/3809.

[26] M. Lesser, "A summary of charge-coupled devices for astronomy," *Publications of the Astronomical Society of the Pacific*, vol. 127, no. 957, p. 1097, Sep. 2015. DOI: 10.1086/684054. [Online]. Available: https://dx.doi.org/10.1086/684054.

[27] A. Ferretti, "Alice upgrades for run 4 and run 5," *Nuclear and Particle Physics Proceedings*, vol. 324-329, pp. 26–29, 2023, QCD 22 is the 25th International Conference on Quantum Chromodynamics, ISSN: 2405-6014. DOI: https://doi.org/10.1016/j.nuclphysbps.2023.01.028. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405601423000354.

[28] N. Bacchetta, P. Collins, and P. Riedler, "Tracking and vertex detectors at fcc-ee," *The European Physical Journal Plus*, vol. 137, no. 2, p. 231, 2022.

[29] G. Aglieri Rinella, "Overview of the ALPIDE pixel sensor features," COMPASS Front-End, Trigger and DAQ Workshop, 2020. [Online]. Available: https://indico.cern.ch/event/863068/contributions/3752479/attachments/1996261/3330551/20200302-Aglieri-ALPIDE-Overview.pdf.

[30] L. S.r.l. "Technology — LFoundry." (2023), [Online]. Available: http://www.lfoundry.com/en/technology (visited on 01/10/2023).

[31] The ARCADIA Collaboration, C. Neubüser, T. Corradino, S. Mattiazzo, and L. Pancheri, "Impact of x-ray induced radiation damage on fd-maps of the arcadia project," *Journal of Instrumentation*, vol. 17, no. 01, p. C01035, Jan. 2022. DOI: 10.1088/1748-0221/17/01/C01035. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/17/01/C01035.

[32] H. A. Bethe, "Passage of radiations through matter," *Experimental nuclear physics*, 1953.

[33] C. F. Powell, P. Fowler, and D. Perkins, *The Study of Elementary Particles by the Photographic Method*. Pergamon Press NY, 1959.

[34] W. H. Barkas, *Nuclear Research Emulsions: I. Techniques and Theory*. Academic Press NY, 1963, vol. 15.

[35] D. A. Glaser, "Some effects of ionizing radiation on the formation of bubbles in liquids," *Phys. Rev.*, vol. 87, pp. 665–665, 4 Aug. 1952. DOI: 10.1103/PhysRev.87.665. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.87.665.

[36] M. Conversi, "The development of the flash and spark chambers in the 1950's," *Le Journal de Physique Colloques*, vol. 43, no. C8, pp. C8–91, 1982.

[37] G. Charpak, R. Bouclier, T. Bressani, J. Favier, and Č. Zupančič, "The use of multiwire proportional counters to select and localize charged particles," *Nuclear Instruments and Methods*, vol. 62, no. 3, pp. 262–268, 1968, ISSN: 0029-554X. DOI: https://doi.org/10.1016/0029-554X(68)90371-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0029554X68903716.

[38] A. Walenta, J. Heintze, and B. Schürlein, "The multiwire drift chamber a new type of proportional wire chamber," *Nuclear Instruments and Methods*, vol. 92, no. 3, pp. 373–380, 1971, ISSN: 0029-554X. DOI: https://doi.org/10.1016/0029-554X(71)90413-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0029554X71904137.

[39] D. R. Nygren, "Proposal to investigate the feasibility of a novel concept in particle detection," *Lawrence Berkeley National Laboratory internal report, Berkeley, CA, USA*, 1974.

[40] "The decay of a lambda particle in the 32 cm hydrogen bubble chamber," 1960. [Online]. Available: https://cds.cern.ch/record/39474.

[41] M. Turala, "Silicon tracking detectors—historical overview," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 541, no. 1, pp. 1–14, 2005, Development and Application of Semiconductor Tracking Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2005.01.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900205000446.

[42] G. Beznogikh, A. Buyak, K. Iovchev, *et al.*, "The slope parameter of the differential cross-section of elastic p-p scattering in energy range 12–70 gev," *Physics Letters B*, vol. 30, no. 4, pp. 274–275, 1969, ISSN: 0370-2693. DOI: https://doi.org/10.1016/0370-2693(69)90438-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0370269369904389.

[43] E. Heijne, P. Jarron, P. Lazeyras, and W. Nelson, "A tiny telescope of si-detectors for high energy muon flux measurement with electron rejection," *IEEE Transactions on Nuclear Science*, vol. 27, no. 1, pp. 272–275, 1980, Cited by: 6. DOI: 10.1109/TNS.1980.4330838. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0018435627%5C&doi=10.1109%5C%2fTNS.1980.4330838%5C&partnerID=40%5C&md5=490d49910060554425267dbb5bc4674b.

[44] G. Bellini, M. Di Corato, P. Manfredi, and G. Vegni, "Live target performances in coherent production experiment," *Nuclear Instruments and Methods*, vol. 107, no. 1, pp. 85–93, 1973, ISSN: 0029-554X. DOI: https://doi.org/10.1016/0029-554X(73)90015-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0029554X73900153.

[45] G. F. Knoll, *Radiation detection and measurement / Glenn F. Knoll*, English, 2nd ed. Wiley New York, 1989, xix, 754 p. : ISBN: 0471815047.

[46] "Semiconductor physics," in *CMOS Electronics*. John Wiley & Sons, Ltd, 2004, pp. 37–52, ISBN: 9780471728528. DOI: https://doi.org/10.1002/0471728527.ch2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471728527.ch2. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471728527.ch2.

[47] F. Hartmann, "First steps with silicon sensors: Na11 (proof of principle)," in *Evolution of Silicon Sensor Technology in Particle Physics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–6, ISBN: 978-3-540-44774-0. DOI: 10.1007/978-3-540-44774-0_2. [Online]. Available: https://doi.org/10.1007/978-3-540-44774-0_2.

[48] A. K. Srivastava, *Si Detectors and Characterization for HEP and Photon Science Experiment How to Design Detectors by TCAD Simulation*, eng, 1st ed. 2019. Cham: Springer International Publishing, 2019, ISBN: 3-030-19531-7.

[49] W.-M. Yao, C. Amsler, D. Asner, *et al.*, "Review of Particle Physics," *Journal of Physics G*, vol. 33, pp. 1+, 2006. [Online]. Available: http://pdg.lbl.gov.

[50] W. Shockley, "Problems related to pn junctions in silicon," *Solid-State Electronics*, vol. 2, no. 1, pp. 35–67, 1961.

[51] R. Caputo, J. S. Perkins, C. A. Kierans, *et al.*, "Astropix: Investigating the potential of silicon pixel sensors in the future of gamma-ray astrophysics," in *Space Telescopes and Instrumentation 2020: Ultraviolet to Gamma Ray*, SPIE, vol. 11444, 2020, pp. 445–456.

[52] R. Bugiel, S. Bugiel, D. Dannheim, *et al.*, "High spatial resolution monolithic pixel detector in soi technology," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 988, p. 164 897, 2021.

[53] H. F.-W. Sadrozinski, A. Seiden, and N. Cartiglia, "4d tracking with ultra-fast silicon detectors," *Reports on Progress in Physics*, vol. 81, no. 2, p. 026 101, Dec. 2017. DOI: 10.1088/1361-6633/aa94d3. [Online]. Available: https://dx.doi.org/10.1088/1361-6633/aa94d3.

[54] G. Aglieri Rinella, "The alpide pixel sensor chip for the upgrade of the alice inner tracking system," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 845, pp. 583–587, 2017, Proceedings of the Vienna Conference on Instrumentation 2016, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2016.05.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900216303825.

[55] Lucas and CMS, *Silicon pixels*, Nov. 2011. [Online]. Available: https://cms.cern/book/export/html/1196.

[56] E. Migliore, "Cms pixel detector design for hl-lhc," *Journal of Instrumentation*, vol. 11, no. 12, p. C12061, Dec. 2016. DOI: 10.1088/1748-0221/11/12/C12061. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/11/12/C12061.

[57] M. Meschini, A. Cassese, R. Ceccarelli, *et al.*, "Radiation resistant innovative 3d pixel sensors for the cms upgrade at the high luminosity lhc," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 978, p. 164 429, 2020, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2020.164429. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900220308263.

[58] G. F. Amelio, "Charge-coupled devices," *Scientific American*, vol. 230, no. 2, pp. 22–31, 1974, ISSN: 00368733, 19467087. [Online]. Available: http://www.jstor.org/stable/24950003 (visited on 10/25/2022).

[59] P. Magnan, "Detection of visible photons in ccd and cmos: A comparative view," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 504, no. 1, pp. 199–212, 2003, Proceedings of the 3rd International Conference on New Developments in Photodetection, ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)00792-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900203007927.

[60] T. Tsuboyama, S. Ono, M. Yamada, *et al.*, "R&d status of soi-based pixel detector with 3d stacking readout," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 924, pp. 422–425, 2019, 11th International Hiroshima Symposium on Development and Application of Semiconductor Tracking Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2018.08.089. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900218310477.

[61] W. Snoeys, G. Aglieri Rinella, H. Hillemanns, *et al.*, "A process modification for cmos monolithic active pixel sensors for enhanced depletion, timing performance and radiation tolerance," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 871, pp. 90–96, 2017, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2017.07.046. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016890021730791X.

[62] Y. Kagawa, S. Hida, Y. Kobayashi, *et al.*, "The scaling of cu-cu hybrid bonding for high density 3d chip stacking," in *2019 Electron Devices Technology and Manufacturing Conference (EDTM)*, 2019, pp. 297–299. DOI: 10.1109/EDTM.2019.8731186.

[63] A. Luštica, "Ccd and cmos image sensors in new hd cameras," in *Proceedings ELMAR-2011*, 2011, pp. 133–136.

[64] A. L. Prasasti, R. K. W. Mengko, and W. Adiprawita, "Vein tracking using 880nm near infrared and cmos sensor with maximum curvature points segmentation," in *7th WACBE World Congress on Bioengineering 2015*, J. Goh and C. T. Lim, Eds., Cham: Springer International Publishing, 2015, pp. 206–209, ISBN: 978-3-319-19452-3.

[65] H. Helmers and M. Schellenberg, "Cmos vs. ccd sensors in speckle interferometry," *Optics & Laser Technology*, vol. 35, no. 8, pp. 587–595, 2003, ISSN: 0030-3992. DOI: https://doi.org/10.1016/S0030-3992(03)00078-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030399203000781.

[66] M. DeLuca, *And not or ccd & cmos technologies in industrial markets*, Nov. 2016. [Online]. Available: http://image-sensors-world.blogspot.com/2016/12/ccd-vs-cmos-market-shares-zoom-needed.html.

[67] D. Dannheim, K. Dort, D. Hynds, *et al.*, "Combining tcad and monte carlo methods to simulate cmos pixel sensors with a small collection electrode using the allpix2 framework," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 2020.

[68] M. J. Pelgrom and M. J. Pelgrom, *Analog-to-digital conversion*. Springer, 2013.

[69] P. R. Barbosa-Marinho, I. Bediaga, F. Barbosa-Ademarlaudo, *et al.*, *LHCb inner tracker: Technical Design Report*, ser. Technical design report. LHCb. Geneva: CERN, 2002, revised version number 1 submitted on 2002-11-13 14:14:34. [Online]. Available: https://cds.cern.ch/record/582793.

[70] T. S. M. C. Limited, https://www.tsmc.com/english/dedicatedFoundry/technology/logic.

[71] X. Llopart, J. Alozy, R. Ballabriga, *et al.*, "Timepix4, a large area pixel detector readout chip which can be tiled on 4 sides providing sub-200 ps timestamp binning," *Journal of Instrumentation*, vol. 17, no. 01, p. C01044, 2022.

[72] L. Pancheri, R. A. Giampaolo, A. D. Salvo, *et al.*, "Fully depleted maps in 110-nm cmos process with 100–300-μm active substrate," *IEEE Transactions on Electron Devices*, vol. 67, no. 6, pp. 2393–2399, 2020. DOI: 10.1109/TED.2020.2985639.

[73] P. D. Group, P. Zyla, R. Barnett, *et al.*, "Review of particle physics," *Progress of Theoretical and Experimental Physics*, vol. 2020, no. 8, p. 083C01, 2020.

[74] M. Intelligence. "CMOS Image Sensor Market Share - Size & Manufacturers." (2023), [Online]. Available: https://www.mordorintelligence.com/industry-reports/global-cmos-image-sensors-market-industry (visited on 08/22/2023).

[75] R. Turchetta, N. Guerrini, and I. Sedgwick, "Large area cmos image sensors," *Journal of Instrumentation*, vol. 6, no. 01, p. C01099, Jan. 2011. DOI: 10.1088/1748-0221/6/01/C01099. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/6/01/C01099.

[76] S. Lauxtermann, "Monolithic deep depletion CMOS pixel sensor for detection of minimum ionizing particles and X rays," 9th International Workshop on Semiconductor Pixel Detectors for Particles and Imaging, PIXEL2018, 2018. [Online]. Available: https://indico.cern.ch/event/669866/contributions/3234993/attachments/1767987/2871451/1100-Sensor_Creations_Pixel_2018_Mono_DD_CMOS_or_MIP_Detection_and_X_Rays.pdf.

[77] K. Aamodt, A. A. Quintana, R. Achenbach, *et al.*, "The alice experiment at the cern lhc," *Journal of Instrumentation*, vol. 3, no. 08, S08002, 2008.

[78] A. Tauro, "ALICE Schematics," General Photo, 2017. [Online]. Available: https://cds.cern.ch/record/2263642.

[79] A. Kluge, "Alice - its3 — a bent, wafer-scale cmos detector," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1041, p. 167 315, 2022, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.167315. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900222006386.

[80] P. Zuccon, "The ams silicon tracker: Construction and performance," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 596, no. 1, pp. 74–78, 2008, Proceedings of the 8th International Conference on Large Scale Applications and Radiation Hardness of Semiconductor Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2008.07.116. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900208010516.

[81] The AMS collaboration, https://ams02.space/.

[82] M. Esposito, C. Waltham, J. T. Taylor, *et al.*, "PRaVDA: The first solid-state system for proton computed tomography," *Phys Med*, vol. 55, pp. 149–154, Nov. 2018.

[83] G. Casse, P. Allport, S. Martì i Garcia, M. Lozano, and P. Turner, "First results on charge collection efficiency of heavily irradiated microstrip sensors fabricated on oxygenated p-type silicon," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 518, no. 1, pp. 340–342, 2004, Frontier Detectors for Frontier Physics: Proceedin, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2003.11.015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900203028353.

[84] F. Baruffaldi, "Development of a proton tomography scanner," Ph.D. dissertation, University of Padova, 2022. [Online]. Available: https://www.researchgate.net/profile/Filippo-Baruffaldi/publication/363781552_Development_of_a_Proton_Tomography_scanner/links/632dcc5586b22d3db4d9b84c/Development-of-a-Proton-Tomography-scanner.pdf.

[85] European Research Council, ERC Consolidator Grants 2014 results, https://erc.europa.eu/sites/default/files/document/file/erc_2014_cog_full_results_by_domain.pdf.

[86] N. Pozzobon, F. Baruffaldi, D. Bisello, *et al.*, "Calorimeter prototyping for the impact project pct scanner," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 936, pp. 1–4, 2019.

[87] S. Mattiazzo, D. Bisello, P. Giubilato, D. Pantano, N. Pozzobon, and W. Snoeys, "The impact project tracker and calorimeter," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 845, pp. 664–667, 2017, Proceedings of the Vienna Conference on Instrumentation 2016, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2016.04.105. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900216303412.

[88] F. Baruffaldi, "Simulations and test beam studies of the impact calorimeter," *Il nuovo cimento C*, vol. 41, no. 3, pp. 1–10, 2018.

[89] The ALICE Collaboration and B. A. et al, "Technical design report for the upgrade of the alice inner tracking system," *Journal of Physics G: Nuclear and Particle Physics*, vol. 41, no. 8, p. 087 002, Jul. 2014. DOI: 10.1088/0954-3899/41/8/087002. [Online]. Available: https://dx.doi.org/10.1088/0954-3899/41/8/087002.

[90] F. Tommasino, M. Rovituso, S. Fabiano, *et al.*, "Proton beam characterization in the experimental room of the trento proton therapy facility," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 869, pp. 15–20, 2017, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2017.06.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900217306654.

[91] G. Tambave, J. Alme, G. G. Barnaföldi, *et al.*, "Characterization of monolithic cmos pixel sensor chip with ion beams for application in particle computed tomography," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 958, p. 162 626, 2020.

[92] B. Przybus, "Xilinx redefines power, performance, and design productivity with three new 28 nm fpga families: Virtex-7, kintex-7, and artix-7 devices," *Xilinx White Paper*, 2010.

[93] M. Da Rocha Rolo, "ARCADIA status report," RD_FCC Meeting with Referees, 2021. [Online]. Available: https://agenda.infn.it/event/27316/contributions/138203/attachments/81991/107620/20210603_ARCADIA_RDFCC.pdf.

[94] A. X. Widmer and P. A. Franaszek, "A dc-balanced, partitioned-block, 8b/10b transmission code," *IBM Journal of Research and Development*, vol. 27, no. 5, pp. 440–451, 1983. DOI: 10.1147/rd.275.0440.

[95] Y.-C. Wu and Y.-R. Jhan, "Introduction of synopsys sentaurus tcad simulation," in *3D TCAD Simulation for CMOS Nanoeletronic Devices*, Springer, 2018, pp. 1–17.

[96] S. Agostinelli, J. Allison, K. a. Amako, *et al.*, "Geant4—a simulation toolkit," *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[97] L. De Cilladi, T. Corradino, G.-F. Dalla Betta, C. Neubüser, and L. Pancheri, "Fully depleted monolithic active microstrip sensors: Tcad simulation study of an innovative design concept," *Sensors*, vol. 21, no. 6, p. 1990, 2021.

[98] C. Neubüser, T. Corradino, G.-F. Dalla Betta, and L. Pancheri, "Arcadia fd-maps: Simulation, characterization and perspectives for high resolution timing applications," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1048, p. 167 946, 2023, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.167946. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900222012384.

[99] "IEEE standard for systemverilog–unified hardware design, specification, and verification language," *IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012)*, pp. 1–1315, 2018. DOI: 10.1109/IEEESTD.2018.8299595.

[100] The ALICE Collaboration, *Letter of intent for alice 3: A next-generation heavy-ion experiment at the lhc*, 2022. arXiv: 2211.02491 [physics.ins-det].

[101] A. Ostrowski and P. Gaczkowski, *Software Architecture with C++: Design modern systems using effective architecture concepts, design patterns, and techniques with C++20*. Packt Publishing, 2021, ISBN: 9781789612462. [Online]. Available: https://books.google.it/books?id=GrAmEAAAQBAJ.

[102] ISO, *Programming language: ALGOL: UDC 681.3.04. Ref. no. ISO/R 1538-1972 (E)*, ser. ISO recommendation. Geneva, Switzerland: International Organization for Standardization, 1972, vol. R 1538, p. 82.

[103] C. Ghabrous Larrea, K. Harder, D. Newbold, *et al.*, "IPbus: a flexible Ethernet-based control system for xTCA hardware," *JINST*, vol. 10, no. 02, p. C02019, 2015. DOI: 10.1088/1748-0221/10/02/C02019.

[104] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[105] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[106] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[107] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*, ser. Prentice Hall electronics and VLSI series. Prentice Hall, 1996, ISBN: 9780133942712. [Online]. Available: https://books.google.it/books?id=DWPwngEACAAJ.

[108] S. Panati, J. Olave, A. Rivetti, *et al.*, "Matisse: A versatile readout electronics for monolithic active pixel sensors characterization," in *2017 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, 2017, pp. 1–4. DOI: 10.1109/NSSMIC.2017.8532806.

[109] M. Ravnik, I. Mele, A. Trkov, J. Rant, B. Glumac, and V. Dimic, "Reactor physics tests of triga mark-ii reactor in ljubljana," ser. Twelfth European TRIGA users conference Papers and abstracts, SPECIFIC NUCLEAR REACTORS AND ASSOCIATED PLANTS, Romania, 2008, p. 286. [Online]. Available: http://inis.iaea.org/search/search.aspx?orig_q=RN:40015670.

[110] D. Žontar, V. Cindro, G. Kramberger, and M. Mikuž, "Time development and flux dependence of neutron-irradiation induced defects in silicon pad detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 426, no. 1, pp. 51–55, 1999, ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(98)01468-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900298014685.

[111] M. Bé and V. Chisté, "F2655e29," in *Monographie BIPM-5—Table of Radionuclides*, vol. 3, 2006.

[112] J. Hubbell and S. Seltzer, "Tables of x-ray mass attenuation coefficients and mass energy-absorption coefficients (version 1.4).[online] available: Http://physics. nist. gov/xaamdi. national institute of standards and technology, gaithersburg," *Originally published as NISTIR*, vol. 5632, 2004.

[113] J. Wyss, D. Bisello, and D. Pantano, "Sirad: An irradiation facility at the lnl tandem accelerator for radiation damage studies on semiconductor detectors and electronic devices and systems," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 462, no. 3, pp. 426–434, 2001, ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(01)00193-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900201001930.

[114] P. Giubilato and W. Snoeys, "Orthopix: A novel compressing architecture for pixel detectors," in *2012 IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC)*, 2012, pp. 1735–1741. DOI: 10.1109/NSSMIC.2012.6551407.

[115] H. Johnson and M. Graham, *High-speed Digital Design: A Handbook of Black Magic*, ser. Prentice Hall Modern Semiconductor Design. Prentice Hall, 1993, ISBN: 9780133957242. [Online]. Available: https://books.google.it/books?id=H5SsQgAACAAJ.

[116] N. H. Weste and K. Eshraghian, *Principles of CMOS VLSI design: a systems perspective*. Addison-Wesley Longman Publishing Co., Inc., 1985.

[117] J. Xue, T. Li, Y. Deng, and Z. Yu, "Full-chip leakage analysis for 65nm cmos technology and beyond," *Integration*, vol. 43, no. 4, pp. 353–364, 2010, ISSN: 0167-9260. DOI: https://doi.org/10.1016/j.vlsi.2010.05.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926010000325.

[118] A. Wiltgen, K. A. Escobar, A. I. Reis, and R. P. Ribas, "Power consumption analysis in static cmos gates," in *2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI)*, IEEE, 2013, pp. 1–6.

[119] G. Timp, J. Bude, F. Baumann, *et al.*, "The relentless march of the mosfet gate oxide thickness to zero," *Microelectronics Reliability*, vol. 40, no. 4-5, pp. 557–562, 2000.

[120] P. Yang, G. Aglieri, C. Cavicchioli, *et al.*, "Low-power priority address-encoder and reset-decoder data-driven readout for monolithic active pixel sensors for tracker system," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 785, pp. 61–69, 2015, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2015.02.063. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900215002818.

[121] E. Serra, M. Angeletti, S. Coli, C. Gargiulo, and R. Iuppa, "Thermo/mechanical design for embedding alpide pixel sensor chip in a high-energy particle detector space module," *Journal of Physics: Conference Series*, vol. 2374, no. 1, p. 012 049, Sep. 2022. DOI: 10.1088/1742-6596/2374/1/012049. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2374/1/012049.

[122] M. Varga-Kofarago, "Medical applications of the alpide detector," *Universe*, vol. 5, no. 5, 2019, ISSN: 2218-1997. DOI: 10.3390/universe5050128. [Online]. Available: https://www.mdpi.com/2218-1997/5/5/128.

[123] P. Sellin, P. Woods, D. Branford, *et al.*, "A double-sided silicon strip detector system for proton radioactivity studies," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 311, no. 1, pp. 217–223, 1992, ISSN: 0168-9002. DOI: https://doi.org/10.1016/0168-9002(92)90867-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0168900292908674.

[124] G. D. Knott, "Hashing functions," *The Computer Journal*, vol. 18, no. 3, pp. 265–278, 1975.

[125] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology—CRYPTO'89 Proceedings 9*, Springer, 1990, pp. 416–427.

[126] A. Mittelbach and M. Fischlin, "The theory of hash functions and random oracles," *An Approach to Modern Cryptography, Cham: Springer Nature*, 2021.

[127] L. Mordell, *Diophantine Equations*, ser. ISSN. Elsevier Science, 1969, ISBN: 9780080873428. [Online]. Available: https://books.google.it/books?id=QugvF7xfE-oC.

[128] G. Contin, L. Greiner, J. Schambach, *et al.*, "The STAR MAPS-based PiXeL detector," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 907, pp. 60–80, Nov. 2018. DOI: 10.1016/j.nima.2018.03.003. [Online]. Available: https://doi.org/10.1016%5C%2Fj.nima.2018.03.003.

[129] P. Kurup and T. Abbasi, *Logic synthesis using Synopsys®*. Springer Science & Business Media, 2012.

[130] M. C. Borja and J. Haigh, "The birthday problem," *Significance*, vol. 4, no. 3, pp. 124–127, 2007.

[131] M. Graphics, *Modelsim 6.5 e reference manual.(2010)*.

[132] E. Ecma, "262: Ecmascript language specification," *ECMA (European Association for Standardizing Information and Communication Systems), pub-ECMA: adr,*, 1999.

[133] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[134] W. R. Inc., *Mathematica, Version 12.0*, Champaign, IL, 2019.

[135] B. W. Kernighan and D. M. Ritchie, *The C programming language*. 2006.

[136] B. Harvey *et al.*, Eds., *C# programming with the public beta*, ser. Programmer to programmer. Chicago, IL, USA: Wrox Press, 2000, pp. ix + 393, ISBN: 1-86100-487-7 (paperback).

[137] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.

[138] T. Page and G. Thorsteinsson, "Routing techniques in vhdl," English, 2007.

[139] H. J. Barnaby, "Total-ionizing-dose effects in modern cmos technologies," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3103–3121, 2006. DOI: 10.1109/TNS.2006.885952.

[140] M. Moll, *Radiation damage in silicon particle detectors. microscopic defects and macroscopic properties*, Dec. 1999.

[141] R. Gaillard, "Single event effects: Mechanisms and classification," in *Soft Errors in Modern Electronic Systems*, M. Nicolaidis, Ed. Boston, MA: Springer US, 2011, pp. 27–54, ISBN: 978-1-4419-6993-4. DOI: 10.1007/978-1-4419-6993-4_2. [Online]. Available: https://doi.org/10.1007/978-1-4419-6993-4_2.

[142] M. Deveaux, G. Claus, G. Deptuch, W. Dulinski, Y. Gornushkin, and M. Winter, "Neutron radiation hardness of monolithic active pixel sensors for charged particle tracking," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 512, no. 1, pp. 71–76, 2003, Proceedings of the 9th European Symposium on Semiconductor Detectors: New Developments on Radiation Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)01878-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900203018783.

[143] "Introduction," in *Fault-Tolerance Techniques for Spacecraft Control Computers*. John Wiley & Sons, Ltd, 2017, ch. 1, pp. 1–27, ISBN: 9781119107392. DOI: https://doi.org/10.1002/9781119107392.ch1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119107392.ch1. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119107392.ch1.

[144] V. A. Gritsenko, "Hot electrons in silicon oxide," *Physics-Uspekhi*, vol. 60, no. 9, p. 902, Dec. 2017. DOI: 10.3367/UFNe.2016.12.038008. [Online]. Available: https://dx.doi.org/10.3367/UFNe.2016.12.038008.

[145] R. Hughes, "Hole mobility and transport in thin sio2 films," *Applied Physics Letters*, vol. 26, no. 8, pp. 436–438, 1975.

[146] D. Fleetwood, "'border traps' in mos devices," *IEEE Transactions on Nuclear Science*, vol. 39, no. 2, pp. 269–271, 1992. DOI: 10.1109/23.277495.

[147] ——, "Fast and slow border traps in mos devices," *IEEE Transactions on Nuclear Science*, vol. 43, no. 3, pp. 779–786, 1996. DOI: 10.1109/23.510713.

[148] R. Lacoe, J. Osborn, D. Mayer, S. Brown, and J. Gambles, "Total-dose tolerance of the commercial taiwan semiconductor manufacturing company (tsmc) 0.35-/spl mu/m cmos process," in *2001 IEEE Radiation Effects Data Workshop. NSREC 2001. Workshop Record. Held in conjunction with IEEE Nuclear and Space Radiation Effects Conference (Cat. No.01TH8588)*, 2001, pp. 72–76. DOI: 10.1109/REDW.2001.960453.

[149] T. Meisenheimer and D. Fleetwood, "Effect of radiation-induced charge on 1/f noise in mos devices," *IEEE Transactions on Nuclear Science*, vol. 37, no. 6, pp. 1696–1702, 1990. DOI: 10.1109/23.101179.

[150] M. Ceschia, A. Paccagnella, A. Cester, A. Scarpa, and G. Ghidini, "Radiation induced leakage current and stress induced leakage current in ultra-thin gate oxides," *IEEE Transactions on Nuclear Science*, vol. 45, no. 6, pp. 2375–2382, 1998. DOI: 10.1109/23.736457.

[151] I. Esqueda, H. Barnaby, and M. Alles, "Two-dimensional methodology for modeling radiation-induced off-state leakage in cmos technologies," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2259–2264, 2005. DOI: 10.1109/TNS.2005.860671.

[152] and and and, "Study of radiation-induced leakage current between adjacent devices in a cmos integrated circuit," *Journal of Semiconductors*, vol. 33, no. 6, p. 064 006, Jun. 2012. DOI: 10.1088/1674-4926/33/6/064006. [Online]. Available: https://dx.doi.org/10.1088/1674-4926/33/6/064006.

[153] M. Moll, "Displacement damage in silicon detectors for high energy physics," *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1561–1582, 2018. DOI: 10.1109/TNS.2018.2819506.

[154] A. Abada, M. Abbrescia, S. S. AbdusSalam, I. Abdyukhanov, and J. Abelleira Fernandez, "Fcc-hh: The hadron collider," *The European Physical Journal Special Topics*, vol. 228, no. 4, pp. 755–1107, Jul. 2019, ISSN: 1951-6401. DOI: 10.1140/epjst/e2019-900087-0. [Online]. Available: https://doi.org/10.1140/epjst/e2019-900087-0.

[155] V. A. J. : v. Lint, *Mechanisms of radiation effects in electronic materials / V. A. J. van Lint ... .et al.*. eng, ser. Wiley interscience publication. New York ?etc?: J. Wiley.

[156] W. de Boer, J. Bol, A. Furgeri, *et al.*, "Radiation hardness of diamond and silicon sensors compared," *physica status solidi (a)*, vol. 204, no. 9, pp. 3004–3010, Sep. 2007. DOI: 10.1002/pssa.200776327. [Online]. Available: https://doi.org/10.1002%5C%2Fpssa.200776327.

[157] R. Wunstorf, "Systematische Untersuchungen zur Strahlenresistenz von Silizium-Detektoren für die Verwendung in Hochenergiephysik-Experimenten," Dr. University of Hamburg, Hamburg, 1992. [Online]. Available: https://bib-pubdb1.desy.de/record/153817.

[158] O. Krasel, C. Gossling, R. Klingenberg, S. Rajek, and R. Wunstorf, "Measurement of trapping time constants in proton-irradiated silicon pad detectors," in *2003 IEEE Nuclear Science Symposium. Conference Record (IEEE Cat. No.03CH37515)*, vol. 1, 2003, 439–443 Vol.1. DOI: 10.1109/NSSMIC.2003.1352079.

[159] A. Chilingarov, "Temperature dependence of the current generated in si bulk," *Journal of instrumentation*, vol. 8, no. 10, P10003, 2013.

[160] D. Campbell, A. Chilingarov, and T. Sloan, "Frequency and temperature dependence of the depletion voltage from cv measurements for irradiated si detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 492, no. 3, pp. 402–410, 2002, ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(02)01353-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900202013530.

[161] W. H. Bell, L. Casagrande, C. Da Via, V. Granata, and V. G. Palmieri, "Temperature dependence of the charge collection efficiency in heavily irradiated silicon detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 435, no. 1-2, pp. 187–193, 1999.

[162] J. Bourgoin, *Point defects in Semiconductors II: Experimental aspects*. Springer Science & Business Media, 2012, vol. 35.

[163] M. Moll, E. Fretwurst, M. Kuhnke, and G. Lindström, "Relation between microscopic defects and macroscopic changes in silicon detector properties after hadron irradiation," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 186, no. 1, pp. 100–110, 2002, ISSN: 0168-583X. DOI: https://doi.org/10.1016/S0168-583X(01)00866-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168583X01008667.

[164] O. Krasel, C. Gossling, R. Klingenberg, S. Rajek, and R. Wunstorf, "Measurement of trapping time constants in proton-irradiated silicon pad detectors," *IEEE Transactions on Nuclear Science*, vol. 51, no. 6, pp. 3055–3062, 2004. DOI: 10.1109/TNS.2004.839096.

[165] G. Lindström, M. Ahmed, S. Albergo, *et al.*, "Radiation hard silicon detectors—developments by the rd48 (rose) collaboration," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 466, no. 2, pp. 308–326, 2001, 4th Int. Symp. on Development and Application of Semiconductor Tracking Detectors, ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(01)00560-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900201005605.