Università degli Studi di Padova

DIPARTIMENTO DI FISICA "G. GALILEI"
Corso di laurea in Magistrale in Fisica

Tesi di Laurea Magistrale

Study and characterization of the APiX sensor for particle tracking applications

Candidato:
Giuseppe SPADONI
Matricola 1106977

Relatore:
Prof. Gianmaria COLLAZUOL

Anno Accademico 2015-2016

A mia sorella.
A mio fratello.
A chi mi vuol bene.

Abstract

A prototype of a new kind of silicon pixel sensor is studied in this work: the APiX (Avalanche PiXel sensor). The device, formed by two vertically-aligned pixel arrays, exploits the coincidence between two simultaneous avalanche events to discriminate between particle-triggered detections and dark counts. A proof-of-concept two-layer sensor with per-pixel coincidence circuits was designed and fabricated in a 150 nm CMOS process and vertically integrated through bump bonding. The sensor includes a 48 x 16 pixel array with 50 μm x 75 μm pixels.

APiX is made of an array of avalanche pixels, that has been designed in a 180 nm CMOS process with high voltage (HV) option. This is a single poly, up to six metal technology. The array includes sensors with a pitch of 50μm$\times$100μm, different size (20μm$\times$20μm, 30μm$\times$30μm and 45μm$\times$43μm) and based on different process layers. Different versions of the front-end electronics, implementing a passive or active quenching technique to suppress the avalanche, have been monolithically integrated in the same substrate as the detector.

The main purpose of this work is to investigate the characteristics of the technology in view of the fabrication of a dual-tier, low material budget sensor for charged particle detection and present the results from the chip characterization in terms of front-end electronics functionality, dark count rate, breakdown voltage, optical cross-talk, quantum efficiency and tracking efficiency with ionizing particles.

Keywords: APD, SPAD, APIX, noise and particles characterization.

Contents

Chapter 1

Introduction

Semiconductor detectors are widely used in particle, nuclear and medical physics. This thesis work focusses on monolithic detectors for tracking and imaging. Pixelated solid state devices can have very high spatial resolutions O(micron) but need quite complicated and costy front-end and readout electronics. These detectors are made by electronically independent units, called pixel. Recently new kinds of pixelated devices for tracking and imaging were proposed which are based on avalanche diode pixels (APD) as elementay units. Here we studied a new device type, based on APD working in geiger mode.

Avalanche photodiodes (APDs) are *p-n*, *p-i-n* o *metal-resistor-semiconductors (MRS)*[20] junction, biased near the breakdown voltage. The I-V characteristic is the same as a normal diode, like figure 1.1. For a short time, before injection of the first carrier into the diode's depletion region, the diode will operate with a voltage above the breakdown voltage, with only leakage current flowing through the junction. The injection of an ionizing carrier into the depletion region creates a self-sustaining avalanche of carriers [8].

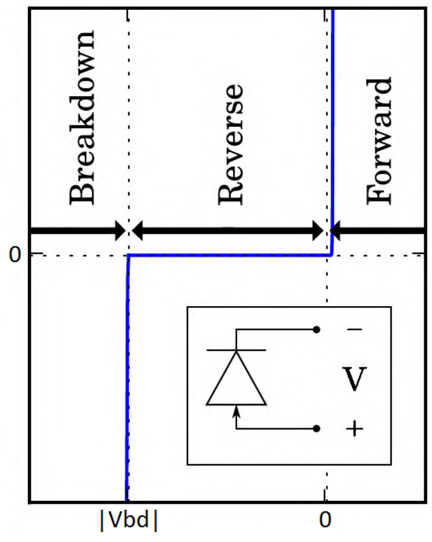


Figure 1.1: Characteristic of a ideal *p-n* junction in steady state. [8]

For single photon measurements avalanche photodiodes can be used. That devices are called Single Photon Avalanche Diodes (SPAD). These are specifically made for operating as single photon detectors. Often, they are developed in silicon [6]. Because of the actual know-how in microelectronics an lithography, the research can improve SPAD performance, using diverse architectures.

The fundamental principle of SPADs is the photo-ionization or charged-particle ionization generation. We can say that when a ionizing radiation enters into the active area of the SPAD, the scattering of this radiation with the semiconductors lattice generates a e-hole couple, which is accelerated due to the high electric field imposed by the external bias voltage. The accelerated drift of carriers produce the impact generation and drifting of new ones, that create a current spike.

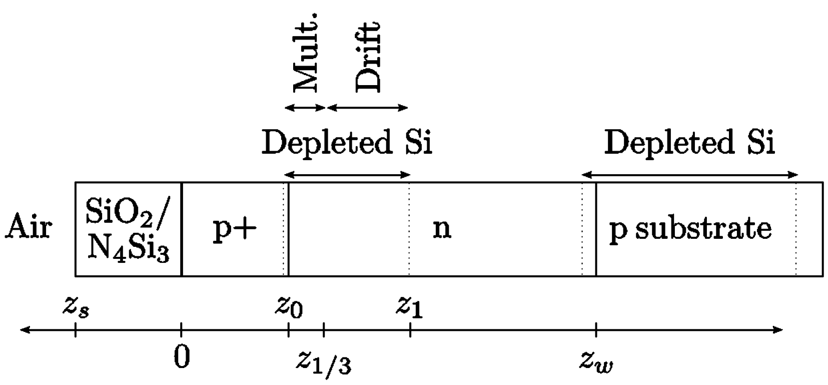


Figure 1.2: Multiplication and drift regions shown on a *p+-n* CMOS chip, with the relevant depletion layer and multiplication and drift regions [8]. The figure is not in scale

This phenomenon is generated into the depletion region. For example in figure 1.2 is shown the section of a p+-n CMOS junction. Due to the high doping level, the depletion region is approximately contained only in the n-doped semiconductor. A SPAD depletion layer can be separated in two parts: the drift region, where the expected carrier generation is negligible; and the multiplication region, where nearly of all of the impact ionization takes place. Qualitatively the multiplication region is defined as the smallest possible region with 95% of carrier generation [8]. Ignoring the depletion layer of p-n junction of the substrate, in figure 1.2 we can note that the depletion layer is extended from z_0 to z_1, the multiplication region is the first third of the layer and the rest is the drift region. Carriers entering from the substrate junction will be swept toward back, and then they not cause ionization since the well-substrate junction's V_{bd} is larger than that of p+-n well due to the smaller doping in the substrate.

The distinction between the drift and the multiplication regions is important. In fact, if the drift region is quite large, the charge is generated uniformly in the volume, creating a so called space-charge-resistance, and introducing a consistent timing response uncertainty.
The size of multiplication region is important for simulations of generation of avalanches and their propagation and, as a consequence, to predict and develop figures of merit of the detector[8].

Although, the main limit of SPADs detectors in the presence of high-frequency noise background. Pancheri et al [11] has develop a new Geiger-mode SPAD detector, called APiX, based on the vertical integration of avalanche, pixels connected in pairs. Those pixels operates in coincidence in fully digital mode and with the processing electronics embedded on the chip. The APiX sensor addresses the need to minimize the material budget and related multiple scattering effects in tracking systems requiring a high spatial resolution in the presence of a large occupancy.

Let now focus on the breakdown process simulation, photo-ionization and charged-particle ionization, analyzing cross sections and features of some simulation and results collected in literature.

1.1 Breakdown simulations

APD and SPAD requires an accurate modeling of semiconductor properties at high electric fields. Fine tuned models for impact ionization and other recombination and generation processes are needed to correctly predict the device breakdown voltage and dark count rate.
Breakdown simulation is necessary to simulate the photon detection probability, by estimating

quantum efficiency and breakdown probability [15].

CMOS SPADs and APDs have a breakdown voltages near 20 V, so the high-field region width is typically in order of $1 \mu m$ or narrower and the peak electric field is larger than $5 \times 10^5 V/cm$. Hayat [9][18] introduced a procedure to estimate the multiplication gain and excess noise factor of avalanche, based on a non-local model. This model was extended to estimate the breakdown probability in Geiger-mode and was applied to devices fabricated in silicon III-V semiconductors.

A series of device simulations were performed for both types of the device. The SPAD's structure for this simulation is simplified. The structure is shown in figure 1.3. The p+/n-well SPAD is also modeled with an asymmetric abrupt junction. As the n-well doping is not constant, this assumption can lead to some innacurancy in the simulation[9].

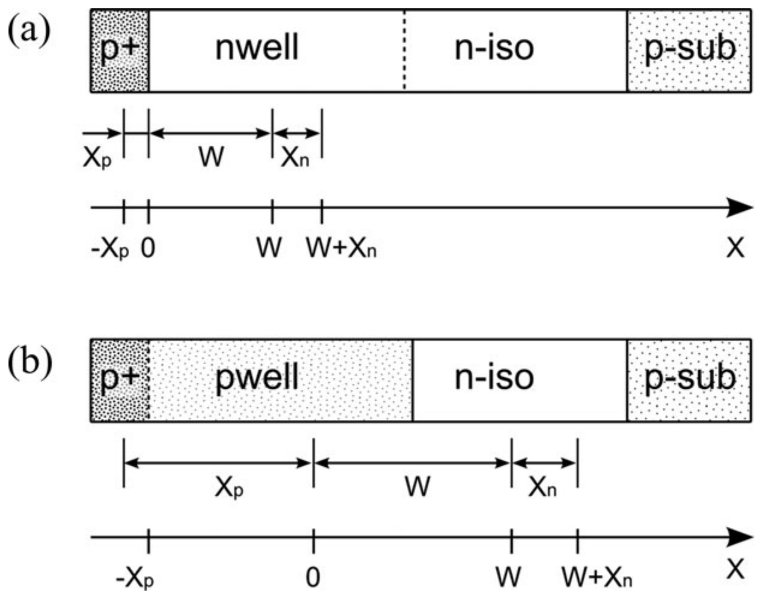


Figure 1.3: Simplified 1-D models used in device simulations[15].

The first quantity simulated is the breakdown probability using the local model. This model focuses in the avalanche generation using a probability density functions of avalanche triggering, one for electrons and one for holes.
An electron located at a position x has a probability $P_e(x)$ to trigger a breakdown event. Similarly, $P_h(x)$ is the probability of holes to trigger an avalanche. The breakdown probability $P_b(x)$ respect the conditionate probability theorem[15][8]:

$$P_b(x) = P_e(x) + P_h(x) - P_e(x)P_h(x) \tag{1.1}$$

For Hayat and McIntyre[8][9][18], the trigger probability can be obtained by a controlled diffusion equation [8][15]:

$$\frac{dP_e}{dx} = -(1 - P_e)\alpha_e[P_e + P_h - P_eP_h], \tag{1.2}$$

$$\frac{dP_h}{dx} = (1 - P_h)\alpha_h[P_e + P_h - P_eP_h], \tag{1.3}$$

where α_e and α_h are the ionization rate for electrons and holes, described in section 1.2. The boundary condition is that $P_e(W) = 0$ and $P_h(0) = 0$. In this case, however, the electric field can exceed the peak of $5 \times 10^5 V/cm$, the upper limit described in the first chapter[22].

The results (figure 1.4) show that the model are in agreement to Okuto's empirical expressions[15] and Fishburn's argumentation, which says that the drifting region the in the first 30% of the total length (red line) of the depletion layer[8].

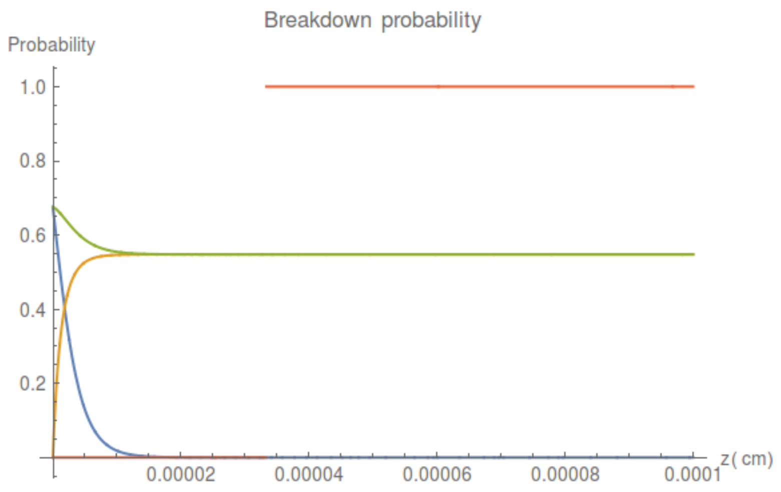


Figure 1.4: Breakdown probability simulation, using parameter from [15], green is for holes and yellow from electrons. The composed probability is the blue line.

The second step of the simulation is to obtain of breakdown probability and breakdown voltage. To obtain this parameter we are using the *dead-space model*, created by McIntyre and used for some structure by Hayat [9]. The APiX case is simulated by Pancheri et al. [15].

In dead-space model, an electron (hole) ha to acquire an energy at least equal to electron (hole) ionization threshold energy of the material E_{ie} (E_{ih}) in order to be able to generate an electron-hole pair by impact ionization. The electron (hole) has to travel a dead-space d_e (d_h) before being able to cause impact ionization. Electron and hole dead spaces are related to the electrostatic potential $\psi(x)$ in the space-charge region using the relation:

$$\psi[x + d_e(x)] - \psi[x] = \frac{E_{ie}}{q} \tag{1.4}$$

$$\psi[x] - \psi[x - d_h(x)] = \frac{E_{ih}}{q}. \tag{1.5}$$

Considering an electron generated in x, the probability density function of the distance ξ to the first impact ionization is $h_e(\xi|x) = 0$ for $\xi < d_e$ and, for $\xi > d_e$:

$$h_e(\xi|x) = \alpha_e(x+\xi)e^{-\int_{d_e(x)}^{\xi} \alpha_e(x+y)dy}. \tag{1.6}$$

Similarly for the hole: $h_h(\xi|x) = 0$ for $\xi < d_h$ and

$$h_h(\xi|x) = \alpha_h(x-\xi)e^{-\int_{d_h(x)}^{\xi} \alpha_h(x-y)dy} \tag{1.7}$$

for $\xi > d_h$. The probability $P_Z(x)$ that an electron generated at x originates a finite number of carriers is related to $P_e(x)$ by the relation:

$$P_Z(x) = 1 - P_e(x), \tag{1.8}$$

and, in the same way for holes, we have $P_Y(x) = 1 + P_h(x)$. in the end the breakdown probability becomes $P_b(x) = 1 - P_Z(x)P_Y(x)$. P_Z and $P_Y(x)$ can be estimated by resolving these integral recursive equations:

$$P_Z(x) = \int_{W-x}^{+\infty} h_e(\xi|x)d\xi + \int_0^{W-x} P_Z^2(x+\xi)P_Y(x+\xi)h_e(\xi|x)d\xi \tag{1.9}$$

$$P_Y(x) = \int_x^{+\infty} h_h(\xi|x)d\xi + \int_0^{x} P_Y^2(x-\xi)P_Z(x-\xi)h_h(\xi|x)d\xi \tag{1.10}$$

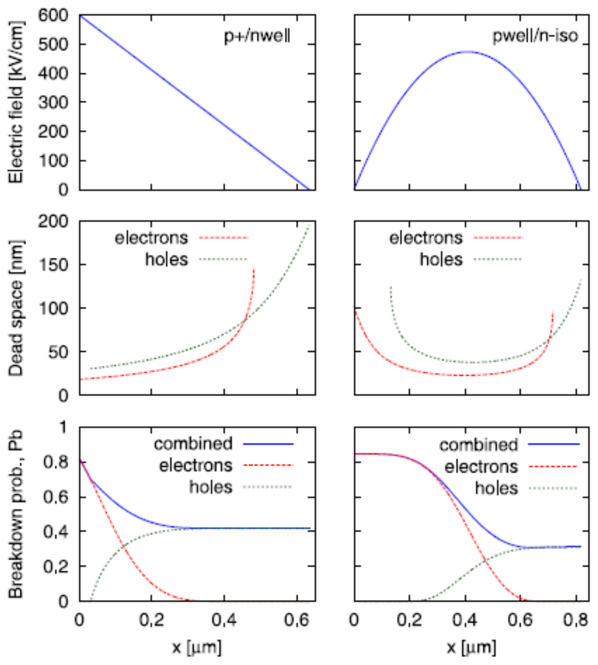


Figure 1.5: Simulated electric field, dead spaces and breakdown probabilities at $V_E = 2V$. Left: p+/n-well. Right: pwell/n-iso[15].

SPAD type	p+/nwell	pwell/n-iso
nwell doping	$6.2 \times 10^{16} cm^{-3}$	
grading coefficient		$3.7 \times 10^{21} cm^{-4}$
p neutral region width $X_P (\mu m)$	0.05	0.63
n neutral region width $X_N (\mu m)$	0.10	0.18
Simulated V_B (V)	16.6	23.7
Simulated temperature profile(V/K)	0.0099	0.0132

Table 1.1: Summary of SPAD model parameters and simulation results[15].

Pancheri et al[15] simulates the breakdown voltage of abrupt and linearly graded junctions with different doping concentration and grading coefficient, in cases of only electron and only hole injection. They obtains good agreement with their experimental data. The parameter used for their simulation and results are summarized in figure 1.5 and table 1.1.

1.2 Impact ionization

Let now focus on the basic phenomenon, which is used for avalanche photodiodes: impact photoionization and charged particle ionization.

First of all let analyze the many-body problem, using Hartree-Fock approximation, and then the application to the diode.

With this brief discussion we want to have the first indication to calculate some of most important figure of merits, that will be discussend into the next chapters.

1.2.1 Many-body problem for photoionization.

This method for the molecules photoionization is theoretical interesting for the application of Hartree-Fock approximation and frozen core one [1], used for many-body fermionic system. This well-know approximation is derived by the mean-field approximation for many-body systems.

Like many body problems, the procedure used to solve it is to find the mean-field potential and the residual interaction potential.
When the interaction equation is find, the solution is find in iterative way, and all the parameters are tabulated by Lucchese et al. [12] [13].
Let start to consider a system composed with general molecules in steady-state, molecules' electrons and an incident photon. The Schrodinger equation is characterized by the exchange-static potential. The electron momentum term and the orbital kinetic term. Of course, the wave function is related to the polarization of input photon and its momentum $\vec{k}$:

$$\left[-\frac{1}{2}\nabla^2 - \frac{1}{r} + V(\vec{r}) - \frac{k^2}{2}\right]\psi_{\vec{k}}^{(\pm)}(\vec{r}) = 0, \tag{1.11}$$

where V is the exchange potential. In the frozen core Hartree-Fock the hyptothesis is that the final state of the electron is described by a single electronic configuration, so an orbital. For this reason the solution is imposed in this form:

$$\psi_{\vec{k}}^{(\pm)}(\vec{r}) = \sqrt{\frac{2}{\pi}}\sum_{l,m} i^l e^{\pm i\sigma_l}\frac{F_l(\gamma;kr)}{kr}Y_{lm}(\Omega_{\hat{r}})Y^*_{lm}(\Omega_{\hat{k}}). \tag{1.12}$$

F_l is determined by an equation of the many-body Columbian problem:

$$\left(-\frac{\nabla}{2} + \frac{Z}{r}\right)\psi_{\vec{k}}(\vec{r}) = \frac{k^2}{r}\psi_{\vec{k}}(\vec{r}), \tag{1.13}$$

and the solution is well-know: in fact, denoted $\rho = k\vec{r}$ and $\eta = \frac{Z}{k}$, the solution is dependent to the function ω:

$$\psi_k(\vec{r}) = \frac{1}{(2\pi)^{2/3}}\frac{1}{r}\sum_{l=0}^{\infty}4\pi(-i)^l\omega(\eta,\rho)Y_l^m(\vec{r})Y_l^{*m}(\hat{k}); \tag{1.14}$$

and, placing 1.14 in 1.13, we obtain the equation:

$$\frac{d^2\omega_l}{d\rho^2} + \left(1 - \frac{2\eta}{\rho} - \frac{l(l+1)}{\rho^2}\right) = 0. \tag{1.15}$$

The final result of this equation is the Columbian function

$$F_l(\eta;\rho) = \frac{2^l e^{-\pi\eta/2}|\Gamma(l+1+i\eta)|}{(2l+1)!}\rho^{l+1}e^{-i\rho}M(l+1+i\eta, 2l+2, \pm 2i\rho), \tag{1.16}$$

where M is the ipergeometrical confluent function [2].

Let return on the original problem and rewrite the equation 1.11 in Lippman-Schwinger for:

$$\psi_k^{(\pm)} = \psi_k^{c(\pm)} + G^{c(\pm)}\psi_k^{(\pm)}, \tag{1.17}$$

[1] The Hartree-Fock frozen core approximation use a particular exchange potential for electrons: where Z is the atomic number, r_i the distance from the origin r_{ij} the mutual electron distances and ψ_j the j-electron wave function.

[2] The ipergeometrical confluent function is the solution of the equation like:

$$z\frac{d^2\omega(z)}{dz^2} - (b-z)\frac{d\omega(z)}{dz} - a\omega(z) = 0.$$

where $G^{c(\pm)}$ is the Coulomb Green's function. Eigenvalues and eigenfunctions are determined considering the previous equation. Of course the solutions founded, for Hartree-Fock approximation is a parametrization of Slater determinant:

$$\psi_k = \frac{1}{\sqrt{2}}|\phi_1\alpha\phi_1\beta...\phi_n\alpha\phi_k\beta| - |\phi_1\alpha\phi_1\beta...\phi_k\alpha\phi_n\beta|; \tag{1.18}$$

$$\delta\psi_k = \frac{1}{\sqrt{2}}|\phi_1\alpha\phi_1\beta...\phi_n\alpha\delta\phi_k\beta| - |\phi_1\alpha\phi_1\beta...\delta\phi_k\alpha\phi_n\beta|, \tag{1.19}$$

minimized using the variational method [13]. At the end, let define the dipolar Matrix in both positional and momentum representation:

$$I^L_{\vec{k},\hat{n}} = \sqrt{|\vec{k}|}\left\langle \psi_k \middle| \vec{r}\cdot\hat{n} \middle| \psi^{(-)} \right\rangle; \tag{1.20}$$

$$I^V_{\vec{k},\hat{n}} = \frac{\sqrt{|\vec{k}|}}{E}\left\langle \psi_k \middle| \vec{\nabla}\cdot\hat{n} \middle| \psi^{(-)} \right\rangle. \tag{1.21}$$

that results can be expanded using atomic armonic functions:

$$I^{L,V}_{\vec{k},\hat{n}} = \sqrt{\frac{4\pi}{3}}\sum_{l,m,n} I^{L,V}_{l,m,n} Y^*_l m(\Omega_{\hat{k}}) Y^*_{ln}(\Omega_{\hat{n}}) \tag{1.22}$$

Lucchese et al shows that the cross section is related to the dipolar matrix and angular distribution of lattice's scattering centers;

$$\frac{d\sigma^{L,V}}{d\Omega_{\hat{k}}} = \frac{\sigma^{L,V}}{4\pi}\left[1 + \beta^{L,V}_{\hat{k}} P_2(cos\theta)\right] \tag{1.23}$$

where $\beta^{L,V}_{\hat{k}}$ represent the asymmetry parameter of distribution of scattering centers, dependent by $\left|I^{L,V}_{\vec{k},\hat{n}}\right|^2$ [13], and $P_2(cos\theta)$ the second order Legendre polynomial of the cosine of the angle between $\hat{k}$ and $\hat{n}$. In the same way we can obtain the differential cross section related to a diffused source:

$$\frac{d\sigma^{L,V}}{d\Omega_{\hat{k}}} = \frac{\sigma^{L,V}}{4\pi}\left[1 + \beta^{L,V}_{\hat{n}} P_2(cos\theta)\right]. \tag{1.24}$$

In particular of silicon [13] there is the transition:

$$([Ne]3s^2 3p^2)SL + h\nu \rightarrow ([Ne]3s^2 3p^1)S'L' + e^-(\vec{k}),$$

using the previous iterative variational method we obtain[13]:

$$\sigma_\nu = \sigma[\alpha x + (\beta - 2\alpha)x^{s+1} + (1 + \alpha + \beta)x^{s+2}] * 10^{-18} cm^2, \tag{1.25}$$

where $x = \frac{\lambda}{2\pi\nu}S'L'$, λ is the photon wavelenght and ν the frequency correlated to ionization threshold. Coefficient α ,β and s are tabulated in [3].

In the exposition of the derivation of the photo-ionization cross-section, there is no hypothesis about the crystalline lattice or the impurity, ect...

For this reason in the next paragraph is analyzed the determination of the ionization rate parameter, used to derivate some features of diodes.

1.2.2 Ionization rates for p-n junction.

Let analyze the problem of the ionization rate of p-n junction.

Firstly we have to give a profile for impurities. Overstraeten et al [22] considers that the profile of impurities is approximate by exponential distribution:

$$N(x) = N_s[exp(-x/\lambda) - 1] := N_s[exp(-z) - 1] \tag{1.26}$$

and the ionization ratio $\gamma = \frac{\alpha_p}{\alpha_n}$ depend on electric field $\vec{E}$ from electron and hole initiated multiplication factor of the diode. As well as, the last hypothesis consists on that the ionization rates α_p and α_n are assumes constants. The profile of the probability of ionization is dependent on the electric field, respecting the Chynoweth's law:

$$\alpha(\vec{E}) = \alpha_\infty e^{-\frac{b}{|\vec{E}|}} \tag{1.27}$$

where α_∞ and b are determined using a properly diode model. Let us consider a reverse biased junction, shown in the figure 1.6. The origin of the x-axis is taken at the junction. The boundaries

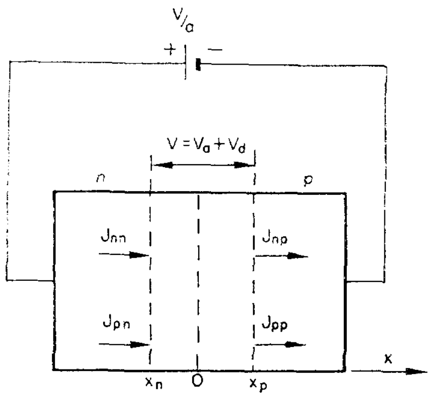


Figure 1.6: Diode model and symbols used for the calculatation of the multiplication factors.

of the depletion layer are respectively x_n and x_p. The total voltage across the junction is:

$$V = V_a + V_d, \tag{1.28}$$

where V_a is the external applied voltage and V_d the build-in ptential. The sign convention for V_a used is that it is positive for reverse bias. The minority carrier currents are referred to as J_{pn}, the hole at x_n and J_{np}, the electron current at x_p respectively. For $qV = q(V_a + V_d)$ much larger than the threshold energy for ionization ϵ_i, the electrons and holes ionize, resulting in an increase of J_{pn} to J_{pp} at x_p and of J_{np} to J_{nn} at x_n. Since for $V_a = 0$ the total voltage across the junction correspond to an energy, qV_d, which is smaller than the threshold energy ϵ_i, there is no ionization.

The multiplication factor at reverse voltage V may be defined and is calculated[22], defining $k = J_{np}/J_{pn}$:

$$M(V) = \frac{J(V)}{J(V_d)} = \frac{J_{nn} + J_{pn}}{J_{pn} + J_{np}} = \frac{J_{pp} + J_{np}}{J_{pn} + J_{np}} =$$

$$= \frac{e^{-\int_{xn}^{xp}(\alpha_n - \alpha_p)dx} + k}{(1+k)1 - \int_{x_n}^{x_p} \alpha_n e^{\int_{xn}^{xp}(\alpha_n - \alpha_p)dx'} dx} = \tag{1.29}$$

$$= \frac{ke^{-\int_{xn}^{xp}(\alpha_n - \alpha_p)dx} + 1}{(1+k)1 - \int_{x_n}^{x_p} \alpha_p e^{\int_{xn}^{xp}(\alpha_n - \alpha_p)dx'} dx}. \tag{1.30}$$

For the superposition principle, we can note that there is a simplest way to calculate the multiplication rates is to calculate that in pure electron or hole injection cases. For pure hole injection $k = 0$, and as a consequence:

$$1 - \frac{1}{M_p(V)} := \Phi_p(V) = \int_{x_n}^{x_p} \alpha_p e^{\int_{x_n}^{x_p}(\alpha_n - \alpha_p)dx'} dx \tag{1.31}$$

and, for pure electron injection ($k \to +\infty$):

$$1 - \frac{1}{M_n(V)} := \Phi_n(V) = \int_{x_n}^{x_p} \alpha_n e^{-\int_{x_n}^{x_p}(\alpha_n - \alpha_p)dx'} dx, \tag{1.32}$$

where $M_n(V)$ and $M_p(V)$ are the multiplication factors, Φ_p and Φ_n the reduced ones. To determinate the ionization rate we have to determinate the diode parameter (x_n, x_p and $\vec{E}(x, V)$) and then the multiplication parameters using the ionization integrals 1.31 and 1.32.

To calculate the electric filed and the boundaries of the depletion layer we have to solve the Poisson equation of this system. A double integration of the Poisson equation yields the following set fo two equation in reduced partial widths z_p and z_n of the depletion layer:

$$\begin{cases} F(z_n) = F(z_p) \\ e^{-z_n}(z_p - z_n - 1) + e^{-z_p} - \frac{(z_p - z_n)^2}{2} = v \end{cases}, \tag{1.33}$$

where[3]:

$$F(z) = e^{-z} + z - 1; \tag{1.34}$$

$$v = \frac{V_a + V_d}{\lambda E_0}; \tag{1.35}$$

$$E_0 = \frac{q\lambda N_s}{\epsilon}. \tag{1.36}$$

Knowing N_s and λ, the parameters of the diode are obtained by numerical solving of 1.33. The electric field is given by:

$$E(z) = E_0[F(z_n) - F(z)] \tag{1.37}$$

and the maximum value of this field is:

$$E_m = E_0 F(z_n). \tag{1.38}$$

Going back to the original coordinate $x = z\lambda$, we obtain the required boundaries of the depletion layer and electric field. The experimental method to derive impurities parameters and V_d is the measurement of the capacitance, discussed later [22].

The determination of ionization ratio γ is determined by equations 1.31 and 1.32. The threshold of ionization impose that the distance $x_p - x_i$ covered by the electron current J_{np} injected into x_p before ionization, referred to figure 1.7, is given by:

$$\psi(x_i) - \psi(x_p) = \int_{x_n}^{x_p} E(x)dx = \frac{\epsilon_i}{q}, \tag{1.39}$$

with ψ is the crystal potential. The holes injected from left to right at x_i generate e-hole pairs by secondary ionization and increase the electron current at x_i. At the and for some manipulation:

$$\gamma = \frac{M_p(V) - 1}{M_n(V) - 1} \tag{1.40}$$

Finally, all parameters of silicon ionization rate are measured and tabulated[22]. This results are used during this work and are showed in figure 1.8. Now we have to focus on how to control the avalanche and the specific use of a SPAD.

[3] q is the electronic charge $q = 1.6 * 10^{-19}C$ and ϵ is the electric permeability of silicon: $1.04 * 10^{-12}F * cm$ [22]

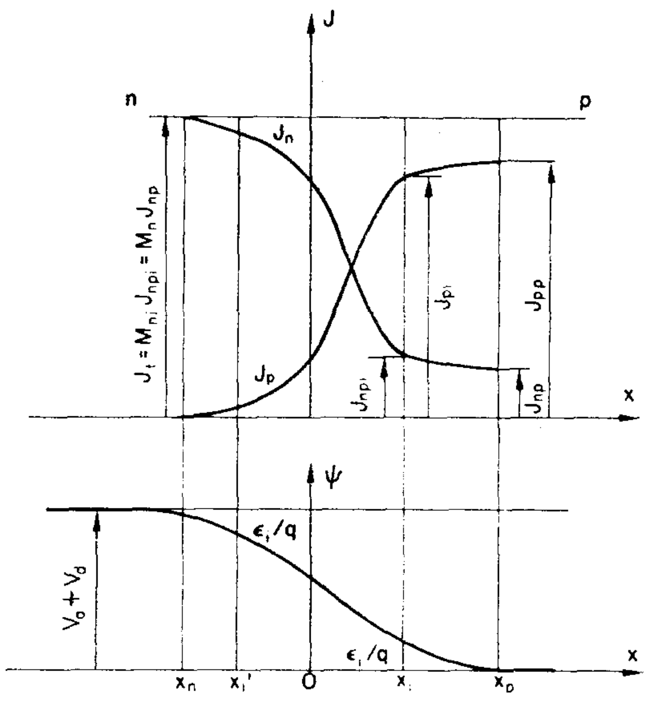


Figure 1.7: Model of the calculation of the influence of the threshold energy ϵ_i on the charge multiplication.

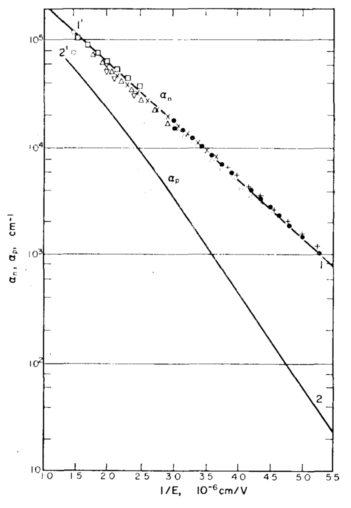


Figure 1.8: Results obtained by Overstraeten et al. for ionization rates α_n and α_p [22].

1.3 Simple circuit implementation: Recharge and quenching

There are some features and parameters to control in SPAD managing, during measurement. First of all we have to control current, because the heat dissipation of the component can damage the diode. Also we have to control the correlated noise, so the applied voltage would be restored rapidly. For this reason there is the necessity of quenching and recharging circuits.

1.3.1 Recharge circuits

For the components chosen, to insert in the box in figure 1.10, which can be used for the recharge, there are two types of recharge circuits: active and passive.
As a passive recharging circuit, we can use a simple resistor. The resistor must be large to

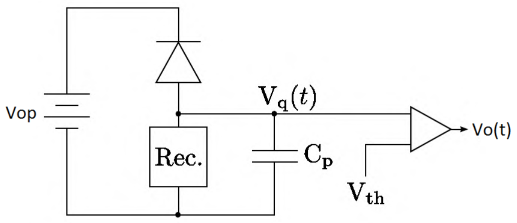


Figure 1.9: General recharge circuit, where the recharge element is represented by the Rec. box. In the figure there is also the comparators output.

challenging the space-charge effect resistance. There are disadvantages with passive circuits, because the restoring time is not brief, consequently the dead time of this detector will become too long. For high events rates an avalanche can occur before the diode is completely recharged [8]. In figure 1.10 we note that the timing distortion becomes if the avalanche does not recharged properly and the bias voltage is not correctly restored .

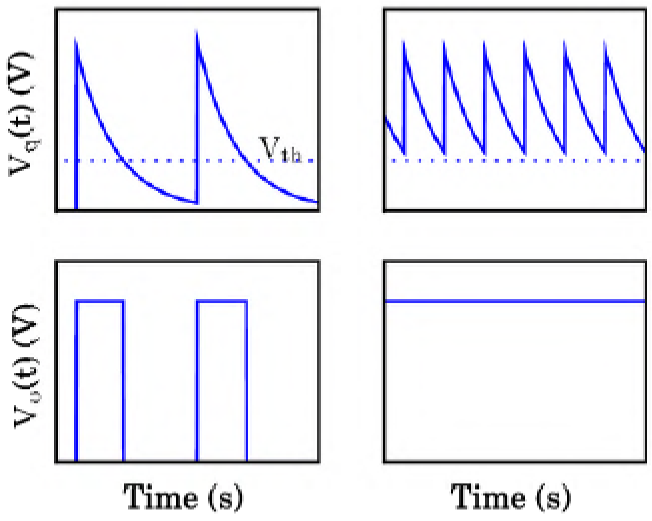


Figure 1.10: A passive recharge circuit normally outputs rising edges, detected by the output if they cross the threshold during the rising (left). If the rate of the events becomes high, the bias voltage cannot be restored, giving a frozen high level (right).

There is also mentioned in literature the possibility of active recharge circuits [8]. The idea is to place a transistor in series with the SPAD use it like an high-impedance switch. The timing responce can be better using a feedback recharge-quenching circuit, creating a monostable element into the detector front-end.
An active recharge gives more saturation behavior when the rising edges in the output pulses are being counted, since the number of pulse will saturate with active recharge, but with passive recharge the number of rising edges will decrease at same point, as figure 1.10 shows.

1.3.2 Quenching circuits.

In addiction to recharge, this architectures also aid in quenching avalanche before completion by sensing avalanche onset and aiding in quenching too.

Circuits that quenches avalanche and the resets the bias voltage plays a key role in SPADs detector performance. Like recharge elements, there are two type of quenching circuits: active and passive.

Passive-quenching circuit (PQC). PCQ is a simple circuit used for the quench. Seeing the model in figure 1.11, the SPAD is reverse biased through a high ballast resistor R_L of 100 kV or more, C_d is the junction capacitance (typically ,1 pF), and C_s is the stray capacitance (capacitance to ground of the diode terminal connected to R_L, typically a few pF). The diode resistance R_d is given by the series of space–charge resistance of the avalanche junction and of the ohmic resistance of the neutral semiconductor crossed by the current. The R_d value depends on the semiconductor device structure: it is lower than 500 V. Avalanche triggering is equivalent

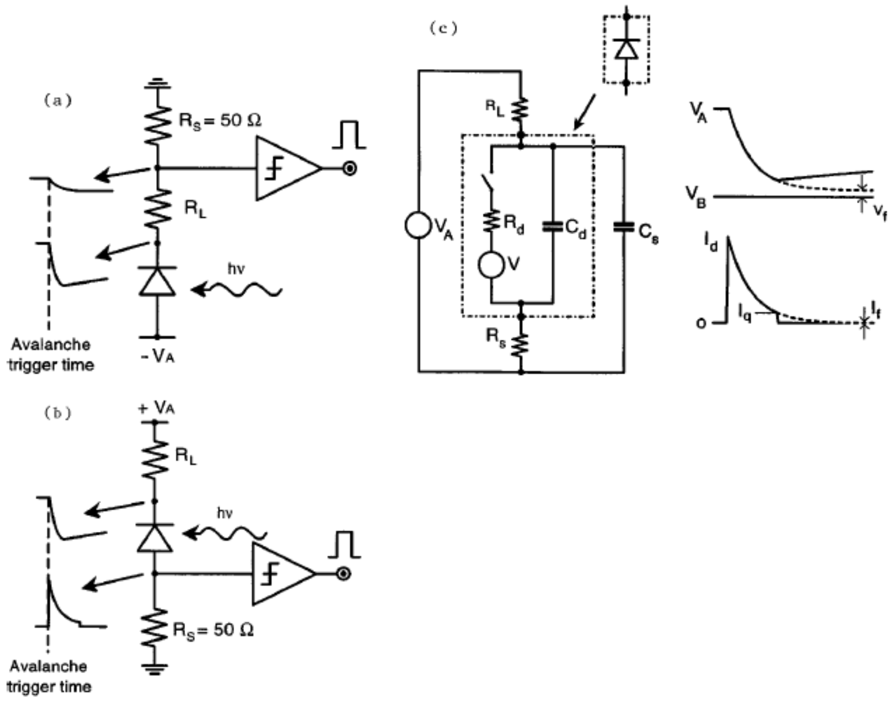


Figure 1.11: Basic PQC's: (a) configuration with voltage-mode output, (b) configuration with current-mode output, (c) equivalent circuit of the current-mode output configuration. The avalanche signal is sensed by the comparator that produces a standard signal for pulse counting and timing.[4]

to close the switch on the circuit in figure 1.11(c) creating a current spike. If $I_d(t)$ and $V_d(t)$ are diode's current and voltage transients:

$$I_d(t) = \frac{V_d(t) - V_{bd}}{R_d} := \frac{V_{ex}(t)}{R_d}. \tag{1.41}$$

When the event is triggered,the avalanche current discharges the capacitances so that V_d and I_d exponentially fall toward the asymptotic steady-state values of V_f and I_f :

$$I_f = \frac{V_A - V_B}{R_d + R_L} \simeq \frac{V_E}{R_L}, \tag{1.42}$$

$$V_f = V_B + R_d I_f. \tag{1.43}$$

The approximation is justified since it must be $R_L \gg R_d$, as shown in the following. The quenching time constant T_q is set by the total capacitance $C_d + C_s$ and by R_d and R_L in parallel, i.e., in practice simply by R_d,

$$T_q = (C_d + C_s)\frac{R_d R_L}{R_d + R_L} \simeq (C_d + C_s)R_d := T_r. \tag{1.44}$$

If I_f is very small, V_f is very near to V_B. When the declining voltage $V_d(t)$ approaches V_B, the intensity of $I_d(t)$ becomes low and the number of carriers that traverse the avalanche region is then small. Since the avalanche process is statistical, it can happen that none of the carriers that cross the high field region may impact ionize.
The probability of such a fluctuation to zero multiplied carriers becomes significant when the diode current I_d falls below <100 μA, and rapidly increases as I_d further decreases[4]. The presence of the parasitic capacitances gives a dead time to recovery system, which is in microsecond range typically. As the recovery starts, the diode voltage V_d rises over V_{bd}. if a photon arrive during the recovery the triggering probability is low, but during the last part of the restoring process, the event triggering becomes possible.
The output pulse has the amplitude:

$$V_u = (V_A - V_B - I_q R_d)\frac{R_s}{R_L + R_s} \simeq V_E \frac{R_s}{R_L} \tag{1.45}$$

A drawback of the voltage-mode output is that the detector timing performance is not fully exploited, because of the intrinsic low-pass filter with time constant T_q that acts on the fast current pulse to produce the voltage waveform.

The energy E_{pd} dissipated in the SPAD during an avalanche pulse corresponds to the decrease of the energy stored in the capacitance $C_d + C_s$:

$$E_{pd} = \frac{1}{2}(C_s + C_d)(V_B + V_E)^2 - \frac{1}{2}(C_d + C_s)V_B^2 \simeq (C_d + C_s)V_E V_B. \tag{1.46}$$

The dissipation therefore depends not only on excess bias voltage V_E and total capacitance $C_d + C_s$, but also on breakdown voltage V_B. At moderate total counting rate n_t, the mean power dissipation is given by $n_t E_{pd}$ and may cause significant heating, particularly in SPAD's with high V_B.
It is worth stressing, however, that PQC's are fairly safe for SPAD's, since they inherently avoid excessive power dissipation. At higher n_t values, the dissipation rise is limited by the increased percentage of small-pulse events; at very high n_t, the limit dissipation corresponds to the product of latching current I_q and breakdown voltage V_B.

Active Quenching circuits(AQC) To avoid drawbacks and damages to SPAD's, active quenching approach is a good solution. The basic idea was simply to sense the rise of the avalanche pulse and react back on the SPAD, forcing, with a controlled bias-voltage source, the quenching and reset transitions in short times. The basic principle is that the rise of the avalanche pulse is sensed by a fast comparator whose output switches the bias voltage source to breakdown voltage V_{bd} or below. After an accurately controlled hold-off time, the bias voltage is switched back to operating level. A standard pulse synchronous to the avalanche rise is derived from the comparator (figure 1.12) output to be employed for photon counting[4]. For a complete exposition we have to say that there is the possibility to create a mixed active-passive quenched circuit, like the example in figure 1.13.

The most important feature of this kind of circuit is the possibility of miniaturization, using CMOS lithography.
In the next part is shown how to create a SPAD CMOS implementation, and a brief analysis of problems and solutions during the construction process.

1.4 CMOS implementation for SPADs: pixel detectors

The implementation in CMOS technology permits to create miniaturized system of SPAD. This technology also permit to create embedded front-end circuitry (recharge and quenching circuits). For this reason, we have the possibility to create a matrix of this fundamental unit, composed by the diode and the front-end circuit, such as a pixel.

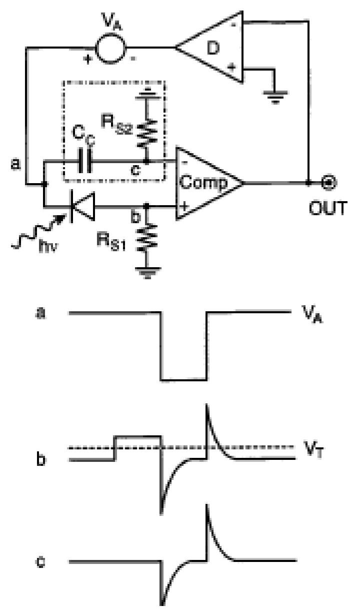


Figure 1.12: Simplified diagram of the basic active quenching circuit configuration with opposite quenching and sensing terminals of the SPAD. The network in the dotted box compensates the current pulses injected by the quenching pulse through the SPAD capacitance, thus avoiding circuit oscillation. The voltage waveforms drawn correspond to the circuit nodes marked with the same letter.[4]

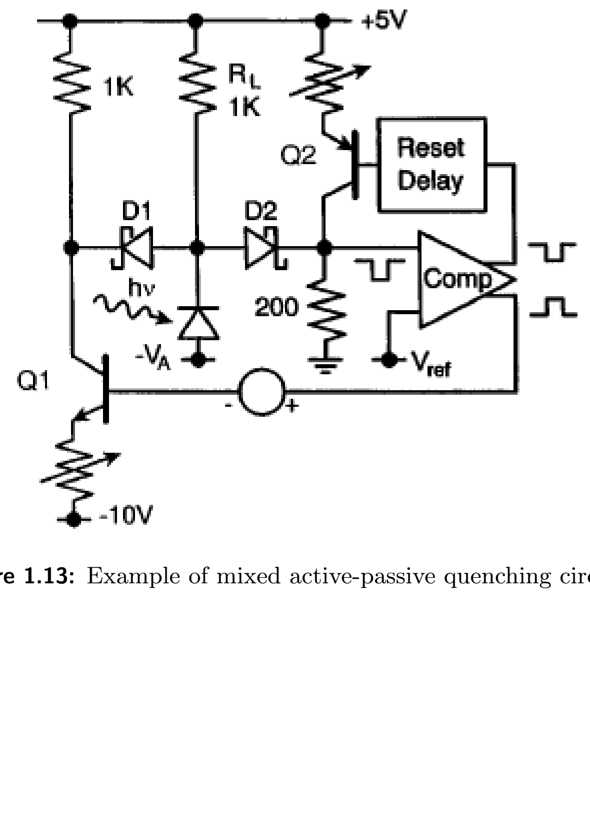


Figure 1.13: Example of mixed active-passive quenching circuit [4].

On the other side, there are two features to manage during the measuring activities and developing. First of all we have to consider the containment of the avalanche into the active area, and then the reduction of the cross-talk between neighbor pixels.
Referred to figure 1.14, when the first avalanche filament is created. The strong carriers concentration gradient drives a lateral diffusion. Few carriers diffuse and in turn starts avalanche multiplication in the sheath surrounding the filament, becouse of the high gradient region outwards. Lateral diffusion then occurs at the new outer edge of the current, and this phenomenon propagates[5]. In the end there is the necessity to separate the active region by the surrounding areas. The structures responsible for this separation is called guard ring. After the implantation

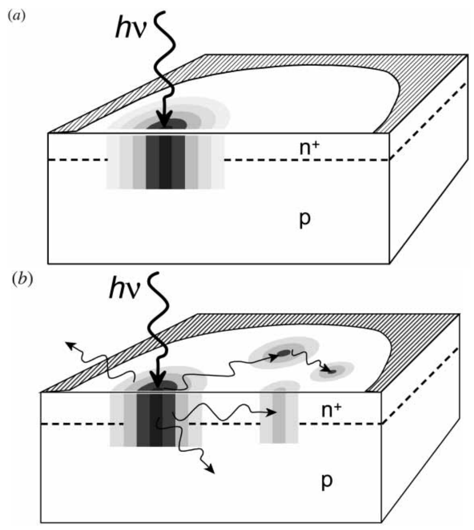


Figure 1.14: Sketch of phenomena that support lateral propagation of the avalanche current started by a photon: (a) lateral diffusion of carriers assisted by avalanche multiplication; (b) absorption within the detector area of photons emitted by hot carriers in the avalanche current.[5]

of doped semiconductor; if the guard ring is not present, the high curvature of the edge will cause the premature breakdown, which reduce the active region. Some SPADs do use *virtual* guard ring, with a look of an extra implant creating doping differences between the structure's outer edge and active region; or *real* guard rings. The guard ring's implementation introduce new constraints in the SPAD design.

The planar SPAD is the only architecture compatible with the standard CMOS processes. Planar SPADs have a depletion layer that is hundreds of nanometer to several micrometer thick [5]. Because most modern CMOS processes use p substrate, a straight-forward SPAD uses the n+ diffusion implant to generate an n+-p junction, with a shallow n-well forming the guard ring. Using a deep n-well isolates SPAD from substrate noise, since the substrate and the deep well form an additional junction that will prevent free carriers in the substrate, which often have a long mean free path, from diffusing into the junction itself. Due to the doping concentration and implant depth in CMOS processes, the depletion layer of implemented SPAD have narrowed and thus a major contribution to noise due to the tunneling. As well as the side effect of high doping levels impose the use of shallow trench isolation (STI)[17] (figure 1.15). STI has had an increasingly important role in the guard ring. The STI include a noise component, due to the trap-filled surface near the depletion region[8]. To reduce this effect, a retrograde junction is used. This junction create a uniform electric field, requiring less physical distance to achieve breakdown.

Dopant in a CMOS process do not have sharp boundaries as often portrayed, and they are not implanted exactly where expected due to the diffusion. The diffusion can cause sizable distortions to a SPAD's multiplication region. In a small diode even distort the expect the value of the diode's breakdown voltage [8].

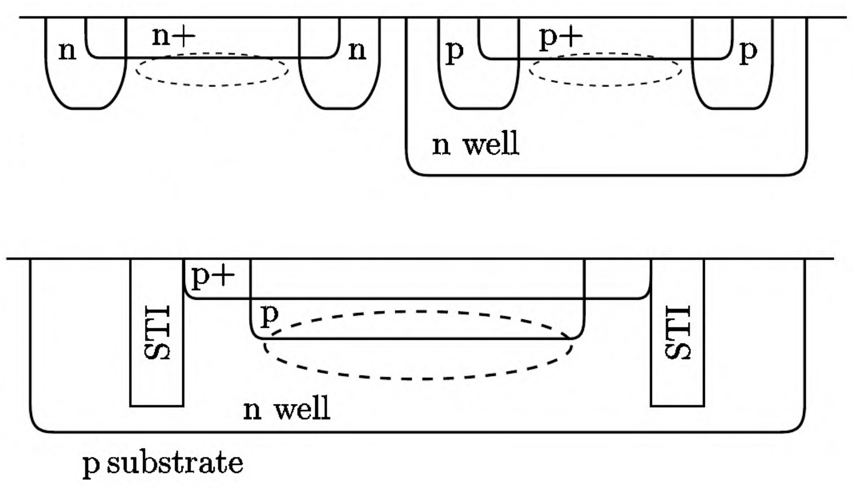


Figure 1.15: (Up)**Cross-sections of SPADs with well-based guard rings**: the multiplication region is highlighted with a dashed eclipse.
(Down)**Cross-section of SPADs with STI-based guard rings**: The multiplication region is highlighted with a dashed eclipse. often, a retrograde n-well or retrograde implant is used to create the junction.

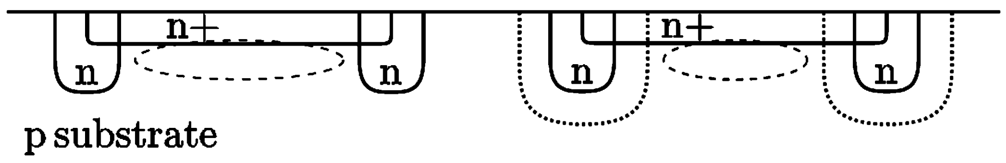


Figure 1.16: The diffusion of well-based guard rings' implant (dotted) can cause sizable distorsion to the multiplication region(dashed).

A feature of CMOS process generation beyond 250-nm is the presence of an optional deep N implant formed by an HV implantation step before n-well formation. This implant is contacted by a ring of n-well and is normally used to completely enclose the p-well regions in order to isolate NMOS transistors from the reminder of the substrate.

Another non-ideal effect consists to the inactive distance in the depletion layer, due to the diffusion of trapped carrier into the STI, reducing the active area, shown in figure 1.16. Due to doped silicon's resistivity, ohmic resistances are introduced between the diode itself and contacts to external circuitry. The ohmic resistance is an important factor when attempting to quickly switch the applied voltage on the diode.

Most of this features are must include to the analysis of the measures during this work, because they create distortions and noise generation.

Chapter 2

The APiX sensor

In actual research at high luminosity LHC, hybrid pixels has been successfully employed in their experiments since their beginning. This type of detector are used because their cheap material budget, low power consumption and radiation hardness[11], because the request of most precise vertex measurement and momentum resolution.

The first solution is to create a thin and full depleted silicon sensors, to reduce multiple scattering. But, in the other hand,this solution directly impacts to signal-to-noise ratio. For this reason is chosen to develop monolithic active pixel sensors, with CMOS technology.

Due to of CMOS foundry capability, there is the possibility to create gain avalanche detectors, that including integrate front-end electronic, with no need of pre-amplification.

The real problem of this kind of devices is represented by dark count rate (DCR). To solve this problem we can use two-tier avalanche pixel detector, consisting of two planar sensors layers. The output signal coincidence is produced when simultaneous strikes both the detectors.
Thanks to cell-to-cell coincidence, the dark count rate DCR_C of a two layer pixel will be reduced to:

$$DCR_C = DCR_1 \cdot DCR_2 \cdot 2\Delta t \tag{2.1}$$

where DCR_1 and DCR_2 are the dark count rates of the two cells, rescectively,and Δt is the coincidence time resolution, so the duration of the signal provided by front-end electronics [14][11].

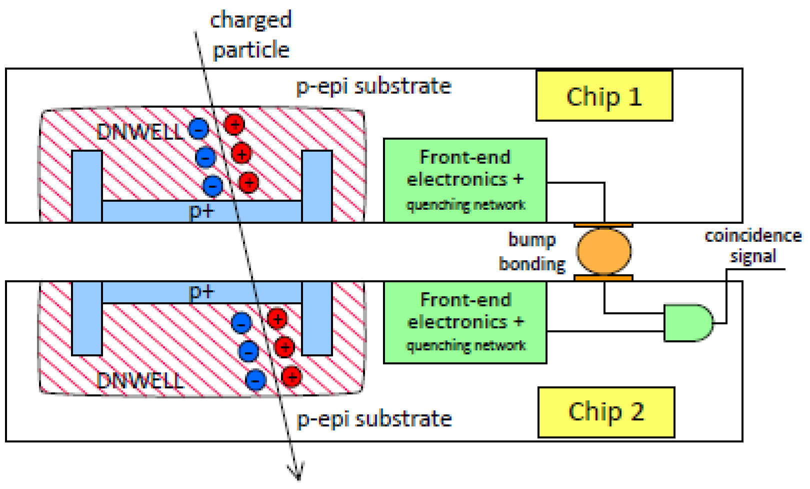


Figure 2.1: Single cell of a dual-layer avalanche pixel detector.

APiX is a dual-layer array of avalanche pixels, which has been developed is 180 nm CMOS process with high voltage option. This technique is usually used for automotive market(figure 2.1).
The chip is comprised of a number of different structures, including single avalanche diodes with

different size and based on different combination of technologies layers, for different type of measures. Moreover, three different of the readout channel have been integrated, one implementing an active quenching technique. In the array, the different options for the sensors and front-end channel are put together so as to cover all possible combinations[14].

Now we analyze the architecture of APiX chip in the first section. In the second one we focus on the procedures, that are used for the enabling and the data acquisition.

2.1 The Architecture

Let we analyze the architecture of the fundamental unit of both chips. APiX is characterized by two different types of Geiger-mode avalanche detectors, represented in figure 2.2. Type 1

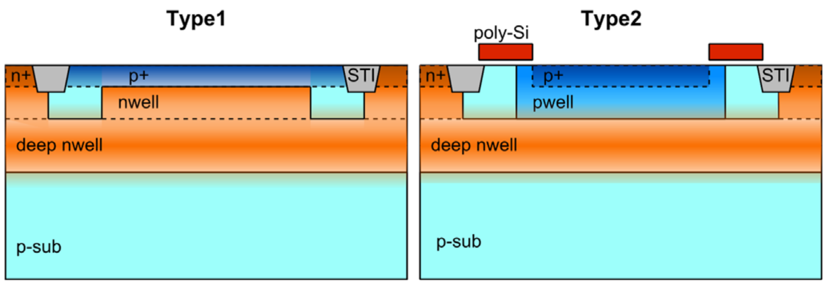


Figure 2.2: Cross section of two detector types.

detector has a shallow p+/nwell junction, while type 2 is based on a deeper pwell/deep nwell junction. The active module is less than 2 μm thick, and both detectors are isolated from the substrate thanks to a deep-nwell.

A simplified pixel block diagram is shown in figure 2.3. The detector is passively quenched

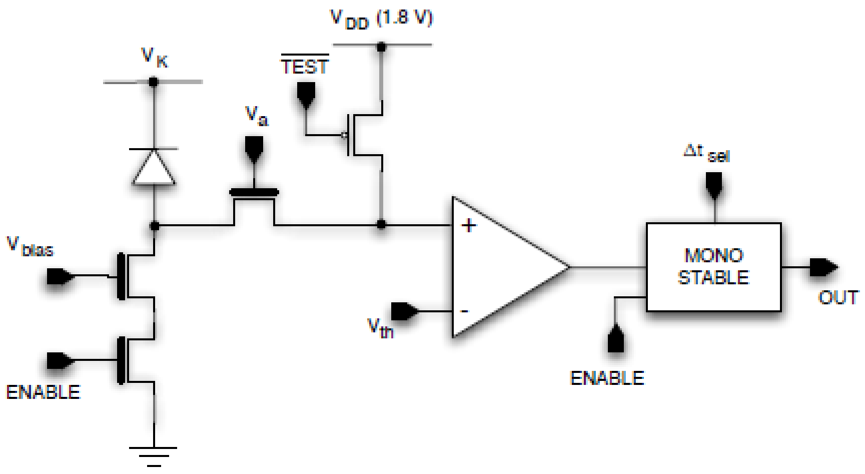


Figure 2.3: Block diagram, with some transistor level detail, of the front-end circuit with passive quenching.

and their output signal are digitalized using a low-threshold comparator.Transistor used in the readout channel are all core devices ($V_{DD} = 1.8V$), except for the current source performing the passive quenching operation and the transistor used to adapt the signal level from the sensor to the comparator, which are I/O devices($V_{DD} = 3V$).
The output pulse can be shortened using a monostable circuit in four different width, from 750 ps to 8 ns, as we can see in figure 2.4. The circuit can be tested using an external $\overline{TEST}$ signal.
The pixel can be independently enable and disabled using a register, with arbitrary pattern.

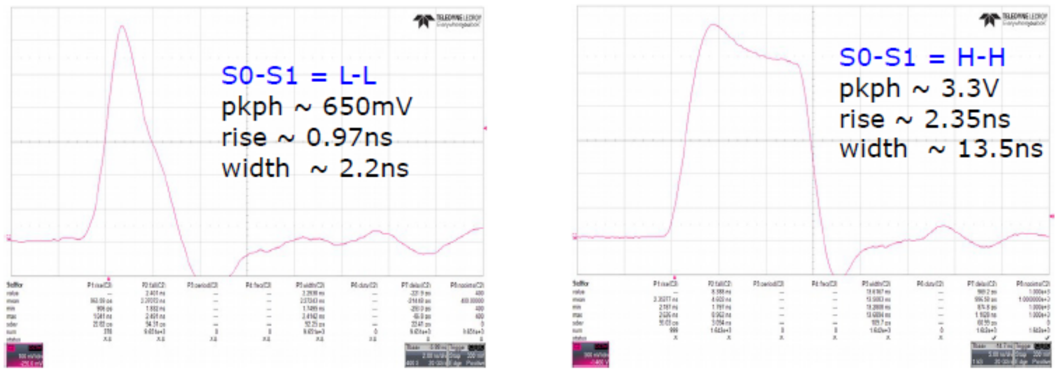


Figure 2.4: APiX C1 monostable outputs visualized using a $1GHz$ probe. The temporal scale is $2ns/div$.

The monostable output signals are also sent to a row-wise OR gate, combining the outputs of all activated pixels. This feature is improved to measure the DCR, that will be analyzed in the next chapter.

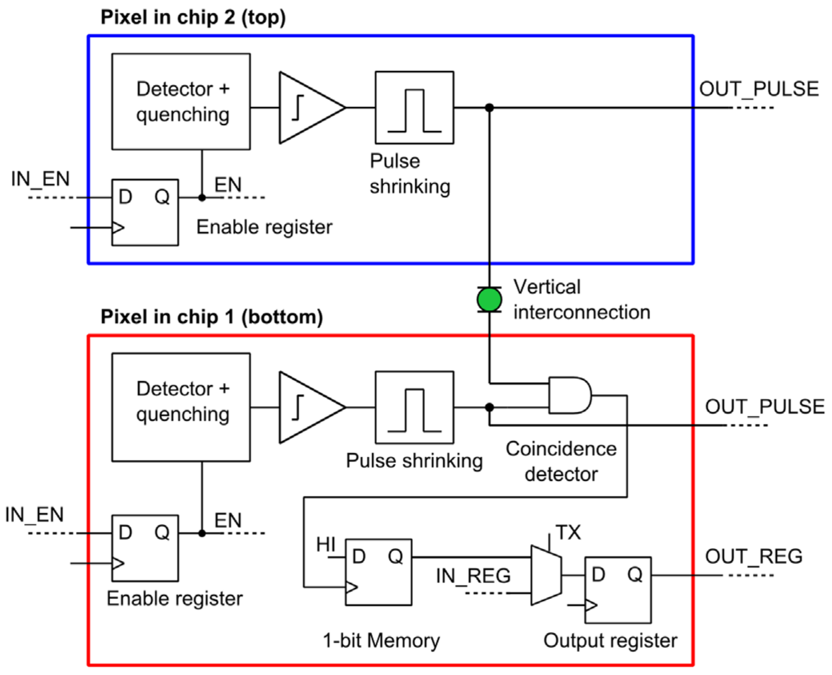


Figure 2.5: Schematic diagram of two layer pixels.[11]

APiX C1 Architecture Referring the red box in figure 2.5, the chip 1 (APiXC1) is composed by a 16×48 pixels array. Each pixel is made by: the avalanche detector and the passive quenching circuit, the 1-bit register for the enabling, the front-end electronics and the output register and the coincidence unit, to connect on the corespondent pixel of the second chip. In addiction to the OR output mentioned before, a row-wise coincidence detection circuit has also been included. The output of two given rows can be connected to the row coincidence detector, as shown in figure 2.6, allowing for the study of the coincidence among arbitrary configuration of pixels between two rows. This feature is used to measure the cross-talk.

APiX C2 Architecture Referring the blue box in figure 2.5, the APiX chip2 (APiXC2) is composed by always a 16×48 array of pixels, that are composed by the detector and quenching circuit, the front-end circuit and the 1-bit enable register. The output of this register is splitted

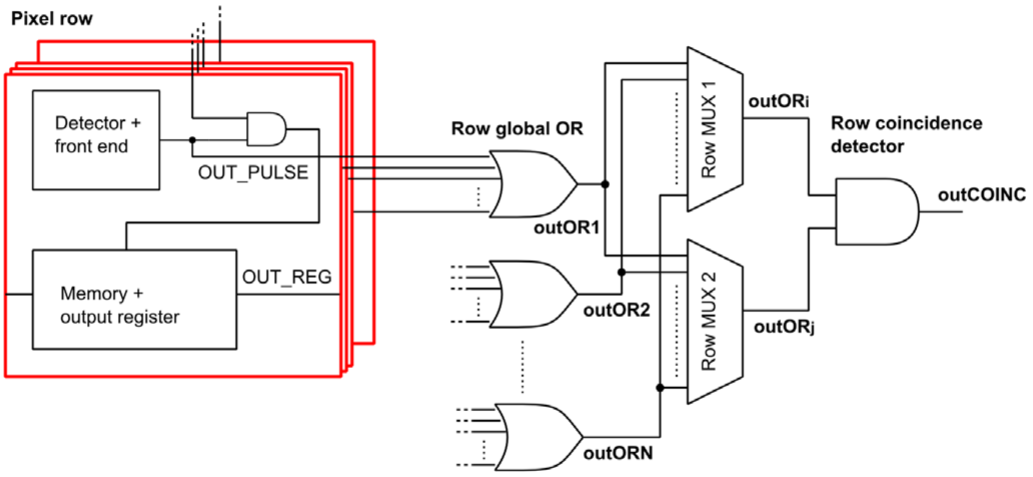


Figure 2.6: Schemating diagram illustrating row-wise coincidence detection unit [11].

in two channels, one of this is connected to to the solder bump, and the other one is used by an independent output for a single-layer measurement.

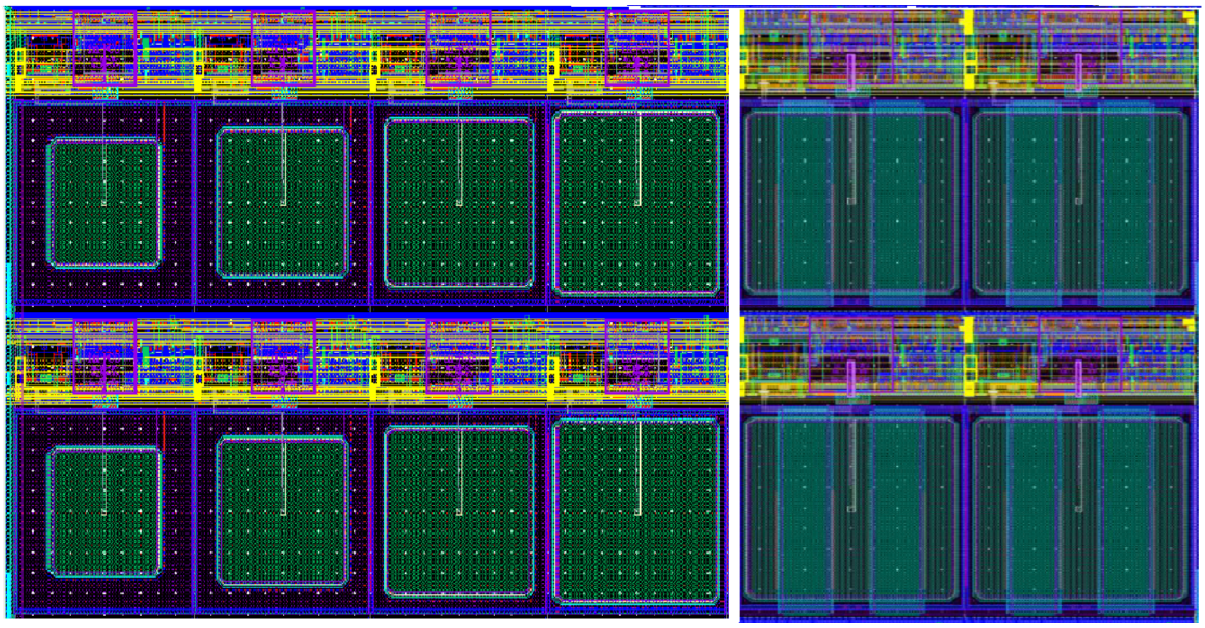


Figure 2.7: Layout of pixels with different size of detectors.[11] On the left there are unshielded pixels and on the right there are the shielded ones.

The core sensor consist of a particular partitioning of the array. Both types of detectors in figure 2.2 were included in the array, in order to choose the best for the application[11]. As we can see in figure 2.7, the array contains detectors with different active areas: $30 \times 30\mu m^2, 35 \times 35\mu m^2, 40 \times 40\mu m^2$ and $45 \times 43\mu m^2$. The main goal of these architecture is to characterize the quantum efficiency of the detector as a function of the filling factor [11].

Finally, the bump bonding technique used a $12\mu m$ solder bumps was chosen for the vertical integration, due to the accessibility and good yield of the process. Most of the pixels were covered with a metal shield to avoid inter-layer optical cross-talk (figure 2.7). A few pixels were left unshielded to study vertical crosstalk and allow for optical measurement.
A summary of arrays partitioning and pixel coupling is shown in figure 2.8.

Let now see the features, functionalists and algorithms necessary to enabling the pattern and to read the data signals.

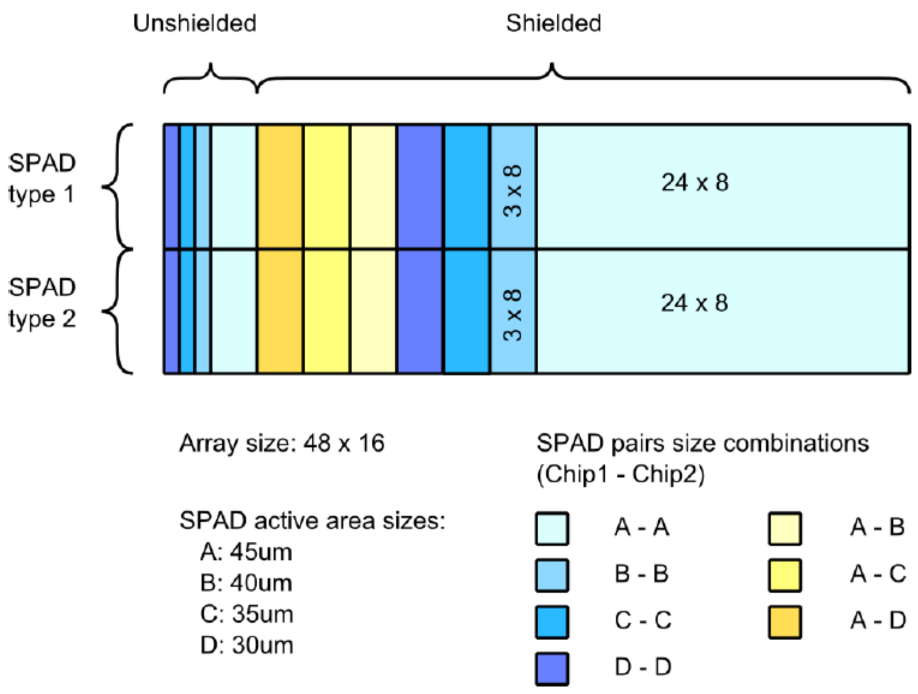


Figure 2.8: Array partition for both chip of APiX detector. The enabling process access single cell from right to left.

2.2 Enabling pixels

The pattern of enabled pixel is independent for the to layers. For this reason, we must have to write twice the procedure. This processes are similar than the write procedure of asynchronous DRAMs.
The signals that are used to apply the pattern on the array is characterized by six digital signals:

- $\overline{TEST\_N}(C2)$: Mentioned below, this signal is used to test the monostable output. This signal is set LOW during all the enabling process. This imposition permits to isolate the SPAD during the programming;

- $\overline{VSR\_RST}(C2)$: this signal reset the content of the vertical shift register, which is use to select rows;

- $VSR\_CLK(C2)$: this signal is use to shift all the rows one by one;

- $\overline{EN\_RST}(C2)$: reset the content of horizontal shift register, so the column selector register;

- $EN\_CLK(C2)$: signal use to shift colums one by one using the horizontal register;

- $EN\_IN(C2)$: bit which is used to activate a single pixel.

Like an asynchronous DRAM, we have to use an electronic system to control the process and to give a clock signal reference. For our experiment we used Arduino DUE (SAM3X8E-ARM) microcontroller, with 10 MHz main clock signal. All procedures are implemented in an Arduino firmware.

Referring the time diagram in figure 2.9, the procedure is used to enabling pixel is:

- Reset the vertical register and put it on the idle state;

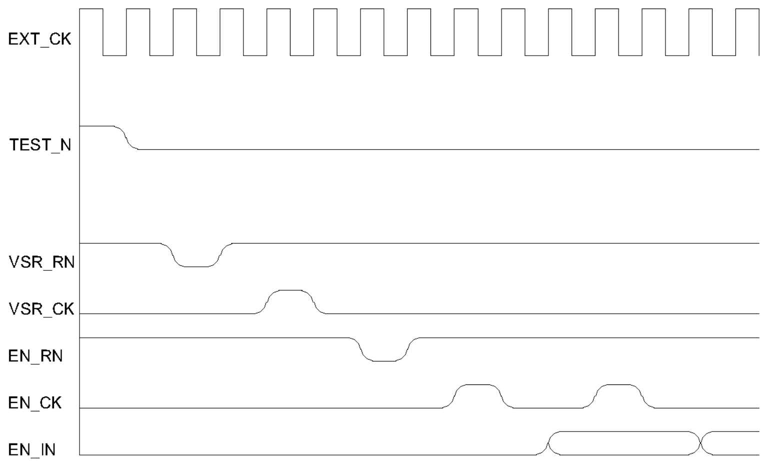


Figure 2.9: Enabling process time diagram.

- Introduce, with a loop, the procedure to shift the VSR. In every iteration a VSR_CLK spike signal is sent to the register. In this way a column was selected and, consequently, the horizontal register is reset to its idle state.
- A second iterative loop was inserted to resolve the horizontal shift register. In this loop a EN_CLK signal is sent to horizontal register. The second operation in the loop consist to set the pattern with an "if" condition. To switch on a pixel is must be necessary to send a EN_IN spike signal from the controller.
- To visualize event in a row, the vertical shift register must be positioned in at the active row position, and send a $\overline{TEST\_N}$ signal.

2.3 Data acquisition

In this last section is introduced the DAQ process. First of all we have to say that there are two read approach: bypassing the FIFO (for both chips) and using the internal memory (only for APiXC1).

In the first case the implementation is simple: the solution is to implement an internal window and count rising edge signal from the chip, and then use an interrupt signal to count OUT_OR rising edges.
A simplified time diagram of this process is shown in figure 2.10, where OUT_OR is the output signal of the detector, count the numeric data and $START\_T$ is the internal signal [1]. A possible result of the APiXC2 scan is shown by the heat-maps in figure 2.11. To manage APiXC1 data acquisition and double-layer's one, the internal memory of the device is used.
The device has eight 96-bit FIFO, one FIFO for two adjacent rows. for this reason the internal signal used are:

- $\overline{FIFO\_RN}$: external FIFO initial reset;
- $\overline{MEM\_RN}$: internal memory initial reset;
- $\overline{FIFO\_SEL}$: external FIFO selector for a 8-bit shift register;

[1]SAM3X8E-ARM microcontroller has four internal PLLs (phase-locked loop), which is used to generate a reference signal with arbitrary frequency.

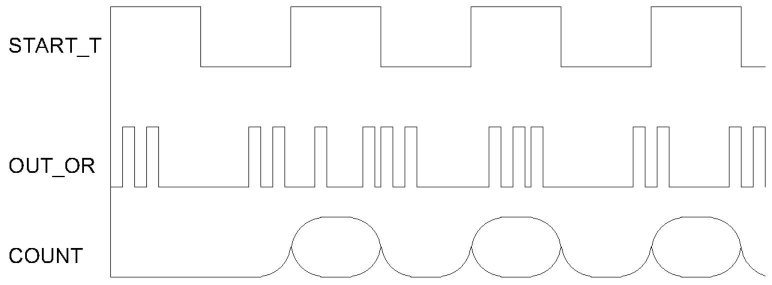


Figure 2.10: Time diagram of C1(bypassing internal memory) and C2 layer data acquisition.

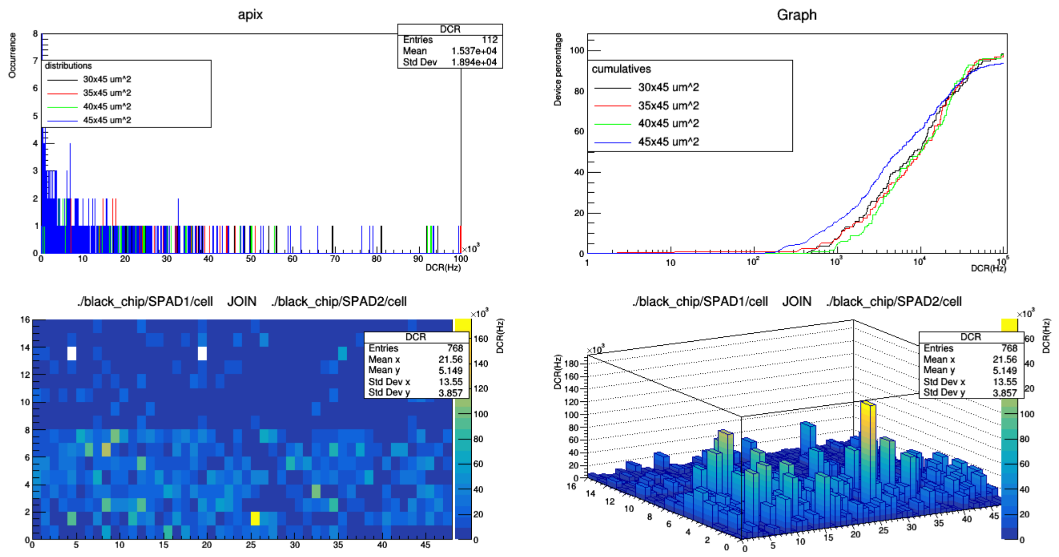


Figure 2.11: APiXC2 heat-maps obtained using the first algorithm. The software used is made ad-hoc using ROOT compiler.

- $FIFO\_TX$: transmission enabling bit;
- $FIFO\_OUT < i >$: This is the output of the i-th FIFO content (from 0 to 7).

In figure 2.12 there is an example of timing diagram of the internal FIFO reading process. In the first part (in the right of the double vertical line) there is the setup reset of both memories, used to erase the undefined data. The communication and the data transfer from $FIFO\_OUT < i >$ is open using a spike of $FIFO\_TX$ signal. This signal is also used to shift the internal FIFO contents (on the left of the dashed line).

This two approaches can be used for different kind of solution, coupled with enable processes. For example we can turn on all the matrix and transfer data from the FIFO or an external memory dump. In the same way we can turn on pixels one by one and read the data using an external process.

Complete firmware algorithms are available on Appendix A.

This solutions are all implemented and discussed in the next chapter, where we will talk of electrical noise features.

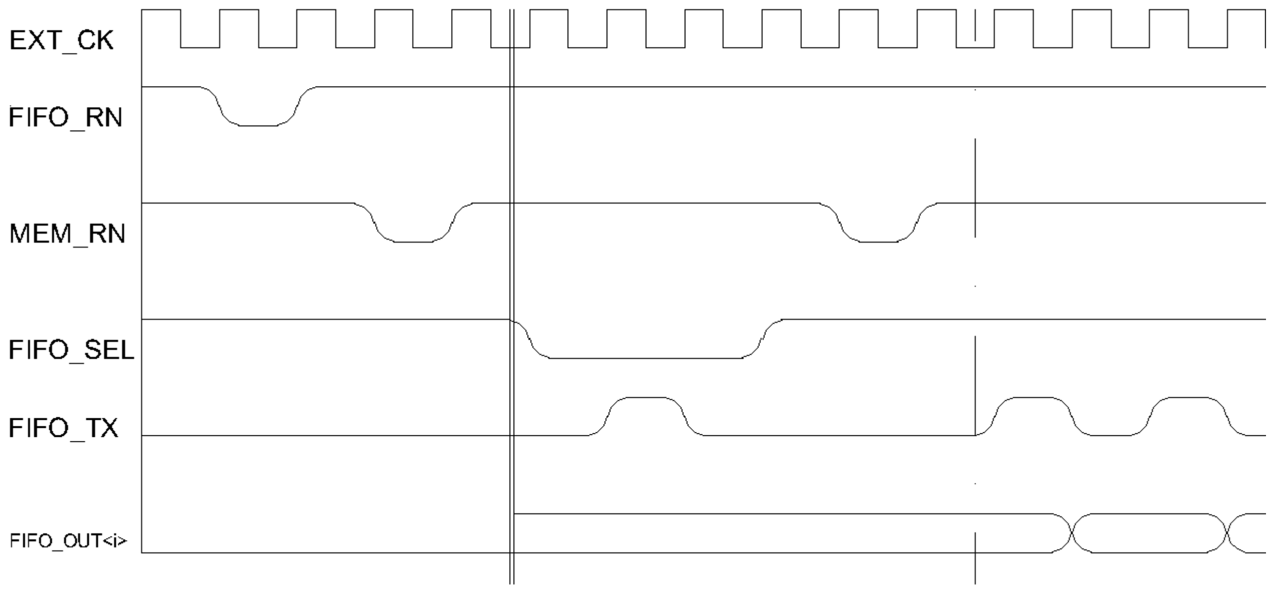


Figure 2.12: Time diagram of the APiXC1 internal FIFO.

Chapter 3

Noise Characterization

Like all electronic instruments, also SPAD are affected by the noise. The noise has two different sources: electrical and optical. Both of this types consists by undesirable injection of carriers.

The dominant components of the noise are after-pulse and cross-talk. Both of they depends on the generation rate of carriers given by specific electric field, as these carriers will cause the spurious avalanches.

In this chapter we focus on the dark count rate and the optical noise, with the determination of some parameters like breakdown voltage,horizontal and vertical cross-talk.

3.1 Dark Count Rate (DCR)

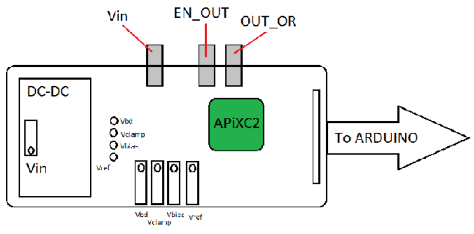
Dark count Rate (DCR) is the rate of events detected in no exposition conditions. The DCR represent the rate of carrier injected into the depletion layer, which are generated in other parts of the device.

For example, the limitation imposed by implants depth, doping concentrations and design rule description in CMOS processes have led to narrow depletion width devices being reported with high DCR due to the tunneling and low photon detection efficiency [17].
Due to the doping concentration and implant depths for the sub 250-nm processes which are required to implement dense digital circuitry, the depletion layer of the SPADs have narrowed and thus a major contributor to dark count has become tunneling. As well as the side effects of high doping levels the use of STI in such processes is know to increase stress and charge traps, thus increasing the DCR and after-pulse[17].

The setups used for DCR characterization are two:

- The first consist on a own-made chip-carrier for APiXC2. Voltage level are regulated by four trimmers. V_{bd} is generated by a DC-DC converter or, optionally, by an external generator. This carrier is connected to an ARDUINO DUE board, which is programmed to enable the array pattern and store data. The external voltage source is gave by ARDUINO board 5V reference, represented in figure 3.1;

- The PCB carrier for APiXC1 and the complete devices is connect to ARDUINO DUE board and level regulation transistor. the voltage references consists by two external generator, which is programmed by a numerical control, written in OCTAVE (or MATlab).
  The chip, the carrier and the ARDUINO board are placed into a climatic chamber, with a range between -50^oC and $+50^oC$. The scheme is in figure 3.2.



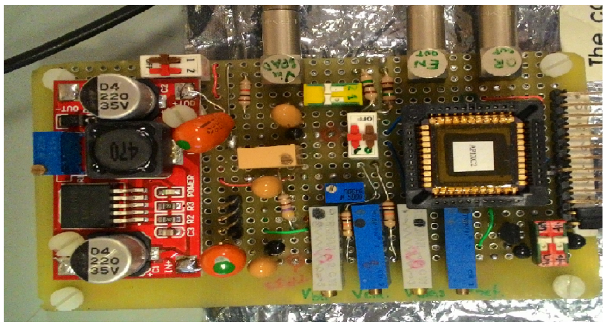
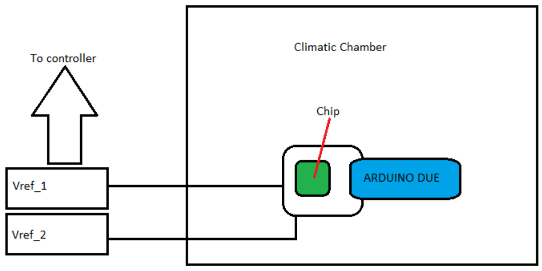


Figure 3.1: (Left) Scheme of the APiXC2 carrier board, with the indication of levels. (Right) photo of the carrier used for the measurements.



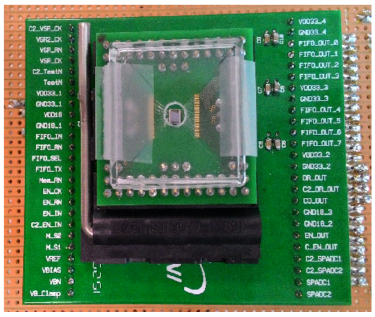


Figure 3.2: (Left) Scheme of the APiXC1 and double layer system, with the indication of levels. (Right) photo of the carrier used for the measurements.

3.1.1 Count-V characteristic and analysis

The first measure we want to discuss is the I-V characteristic of SPAD includes into the array. Due to the design of the pixel, this operation is not possible, but we can do an alternative measure.

This alternative operation consist to acquire dark counts in function of the input voltage level V_{SPAD}.

Using the chip carrier in figure 3.1, we had set manually the voltage reference, and then we had registered some frames at 10 fps for ten seconds. We have choose for both types of SPADs eight cells, four shielded and four unshielded, with different active areas. The voltage applied voltage reference are:

$$V_{REF} = (0.732 \pm 0.001)V \tag{3.1}$$
$$V_{BIAS} = (1.507 \pm 0.001)V \tag{3.2}$$
$$V_{CLAMP} = (2.497 \pm 0.001)V \tag{3.3}$$
$$V_{BN} = (1.500 \pm 0.001)V; \tag{3.4}$$

In all cases we found the same evolution of the DCR, which underline the three regimes: linear, sub-Geiger and Geiger mode, as we can see in figure 3.3. In figure 3.4 we can see that the behavior of the SPAD is the same. Of course some parameters change due to the material. In fact the first parameter is the minimum potential required to permits the avalanche into the SPAD: the **breakdown voltage**.

To measure the breakdown voltage in V_{bd} for APiXC2 cell a linear regression is applied on the linear region of all cells, then the breakdown voltages are extrapolated by the fit, imposing that $DCR = 0$. Results are reported in tables 3.1 and 3.2.

The same characterization is made for APiXC1 device, but with different procedure, because the different available setup.

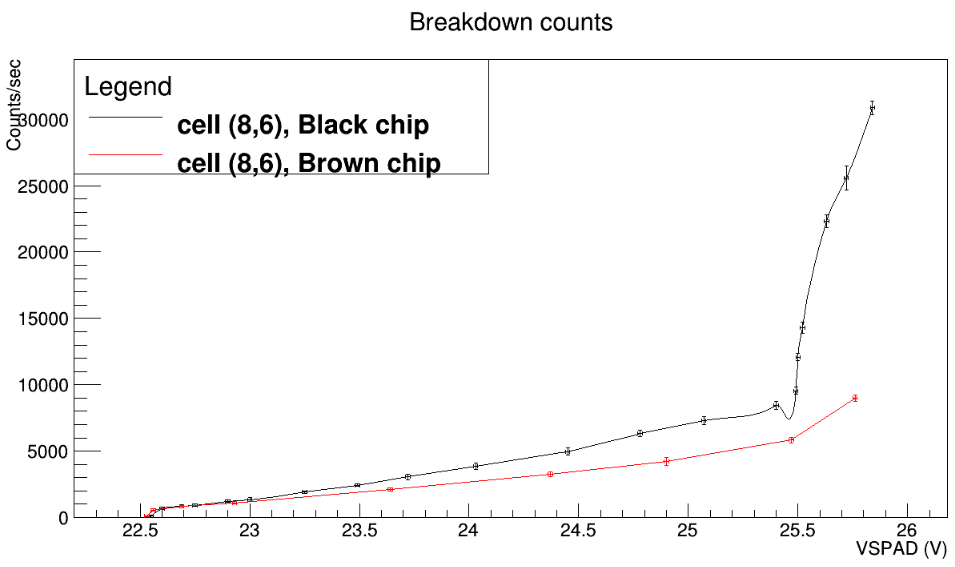


Figure 3.3: Example of two homologue cells of two different chips. We can recognize the linear regime between 22.5 and 25 volts, the sub-Geiger regime at 25.5 volts, and then we found the Geiger regime.

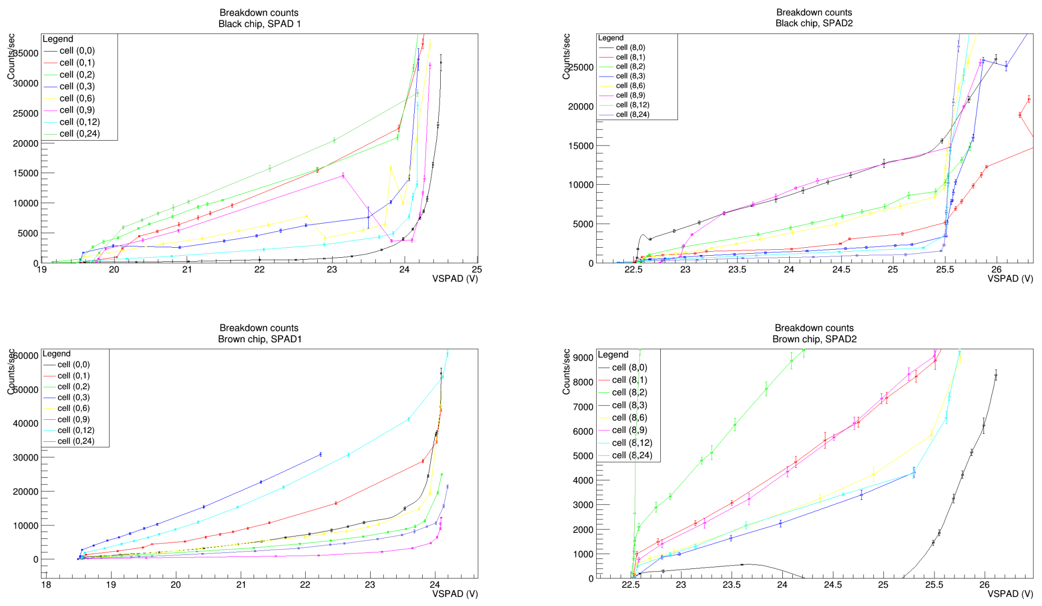


Figure 3.4: DCR-V plot of diverse APiXC2 cells of both types and shielding. There are the results made by using two different chips, named with the color of their support.

With an external controller the V_{spad} voltage is managed by a computer software, using MATlab. Then automatically the frame are register by the help of an external counter.

The result is a little bit different than that obtained with the APiXC2, probably because the better quality of the APiXC1 carrier manufacture. As well as, seeing results in figure 3.5 the linear interpolation of linear regime's point in not possible. To fit the breakdown parameters the error function is used:

$$y = a \int_0^{V_{SPAD}} e^{-\frac{(t-V_{bd})^2}{b}} dt, \tag{3.5}$$

where a, b and V_{bd} are fit parameters. The obtained results are tabulated in table 3.3. The enumeration of APiXC1 cells is different than APiXC2 because the different algorithm used.

We can note that the breakdown voltage, so the working points, are not different for two

Cell coordinate	side length (μm)	type	V_{bd} (V)
SPAD 1			
(0,47)	30	unshielded	(19.22 ± 1.71)
(0,46)	35	unshielded	(19.44 ± 0.78)
(0,45)	40	unshielded	(20.27 ± 0.64)
(0,44)	45	unshielded	(19.72 ± 2.31)
(0,41)	30	shielded	(19.11 ± 1.67)
(0,38)	35	shielded	(19.28 ± 1.42)
(0,35)	40	shielded	(19.22 ± 2.33)
(0,24)	45	shielded	(19.71 ± 1.50)
SPAD 2			
(8,47)	30	unshielded	(21.92 ± 0.87)
(8,46)	35	unshielded	(21.82 ± 0.85)
(8,45)	40	unshielded	(22.46 ± 0.42)
(8,44)	45	unshielded	(22.34 ± 0.89)
(8,41)	30	shielded	(22.50 ± 0.67)
(8,38)	35	shielded	(21.99 ± 2.29)
(8,35)	40	shielded	(22.23 ± 1.32)
(8,24)	45	shielded	(22.01 ± 2.46)

Table 3.1: Black chip's V_{bd} for different cells.

Cell coordinate	side length (μm)	type	V_{bd} (V)
SPAD 1			
(0,0)	30	unshielded	(18.39 ± 0.40)
(0,1)	35	unshielded	(19.08 ± 1.00)
(0,2)	40	unshielded	(18.56 ± 0.86)
(0,3)	45	unshielded	(18.48 ± 0.43)
(0,6)	30	shielded	(18.48 ± 0.43)
(0,9)	35	shielded	(18.40 ± 2.18)
(0,12)	40	shielded	(18.27 ± 0.92)
(0,24)	45	shielded	(18.44 ± 1.28)
SPAD 2			
(8,0)	30	unshielded	(22.19 ± 2.19)
(8,1)	35	unshielded	(22.28 ± 0.87)
(8,2)	40	unshielded	(22.17 ± 0.55)
(8,3)	45	unshielded	(22.28 ± 1.65)
(8,6)	30	shielded	(22.16 ± 2.07)
(8,9)	35	shielded	(22.37 ± 1.18)
(8,12)	40	shielded	(22.20 ± 1.18)
(8,24)	45	shielded	(21.91 ± 0.35)

Table 3.2: Brown chip's V_{bd} for different cells.

chips, the only difference is correlated by the SPAD type. The difference between the results of two arrays consist by the absolute error, that derives by the measuring method. The APiXC2 carrier permit to control the V_{SPAD} voltage using a voltmeter, with a precision of 0.01 volts, but the internal controls used for the APiXC1 carrier gave a better precisions of voltage (0.001 volts). Also the contact between APiXC2 and its carrier sometimes becomes only capacitive, and not resistive. In some cases counts fell down or becomes too higher, because the RTS (random telegraph signal) that hibernate the output status at high level.

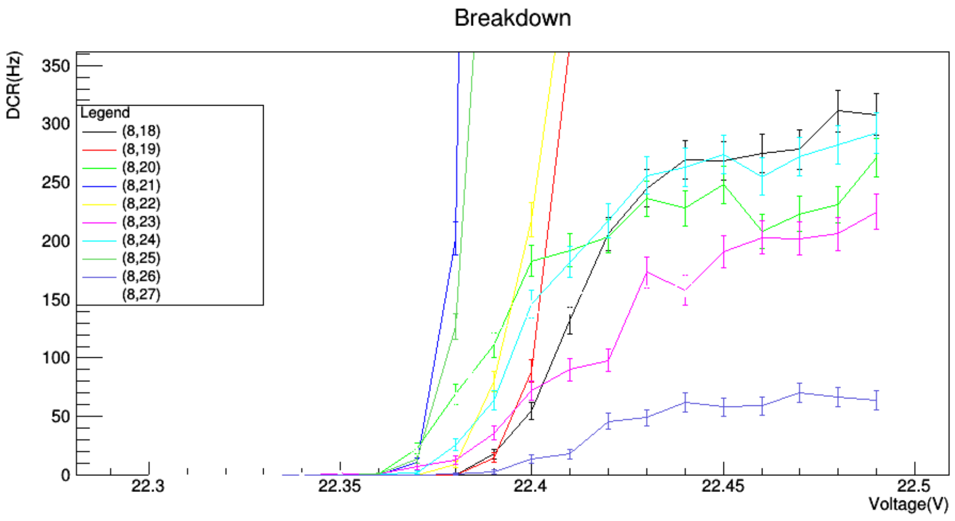


Figure 3.5: Breakdown curves of some pixel of APiXC1's SPAD2.

Cell coordinate	side length (μm)	type	V_{bd} (V)
SPAD 1			
(0,23)	30	shielded	(18.274 ± 0.002)
(0,24)	35	shielded	(18.295 ± 0.008)
(0,27)	40	shielded	(18.267 ± 0.001)
(0,30)	45	shielded	(18.276 ± 0.004)
SPAD 2			
(8,23)	30	shielded	(22.497 ± 0.005)
(8,24)	35	shielded	(22.483 ± 0.004)
(8,27)	40	shielded	(22.490 ± 0.004)
(8,30)	45	shielded	(22.489 ± 0.002)

Table 3.3: APiXC1's V_{bd} for different cells.

The next question is that: there are some parameters that correlated by breakdown voltage? To answer to this question we tried to correlate the breakdown point to active area and environment temperature.

The first analysis is made for the APiXC2. Seeing the figure 3.6 we can note that the breakdown voltage not depends to the pixel area, because the constant profile of V_{bd}.

The relation between temperature and breakdown voltage is made using a bonded chip into the climatic chamber. Both chips are turned on individually. In this way the noise from the other chip is excluded.

The mean value of V_{bd} of the different SPADs is calculated (we can do this operation because the V_{bd} is independent from the area) for different temperatures steps, between -50 and $+30$ ^oC. The results obtained are represented in figure 3.7.

We note intuitively that there is a linear relation between temperature and noise events, and the behaviour is also independent to the active area. The fit results are tabulated in table 3.4. Also the behavior is the same for both chips and are compatible with the error. The main argument is that the temperature control has a key role into a detecting measurement, to control the noise events.

We can note that our results according to simulations showned in the first chapter(table 1.1), but, for both SPADs, the dead-space model gives an underestimated breakdown probability.

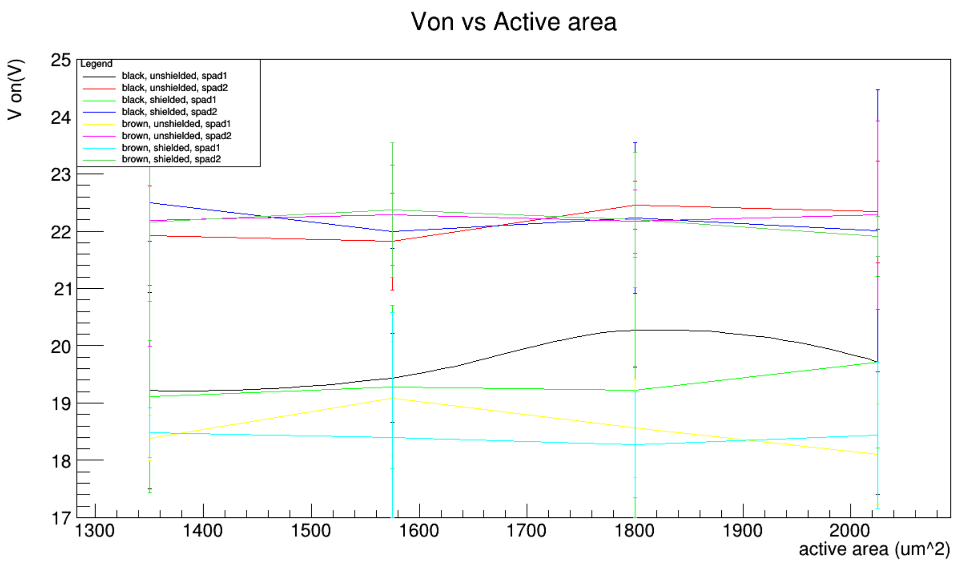


Figure 3.6: V_{bd} vs temperature for cells of APiXC2, using the board in figure 3.1

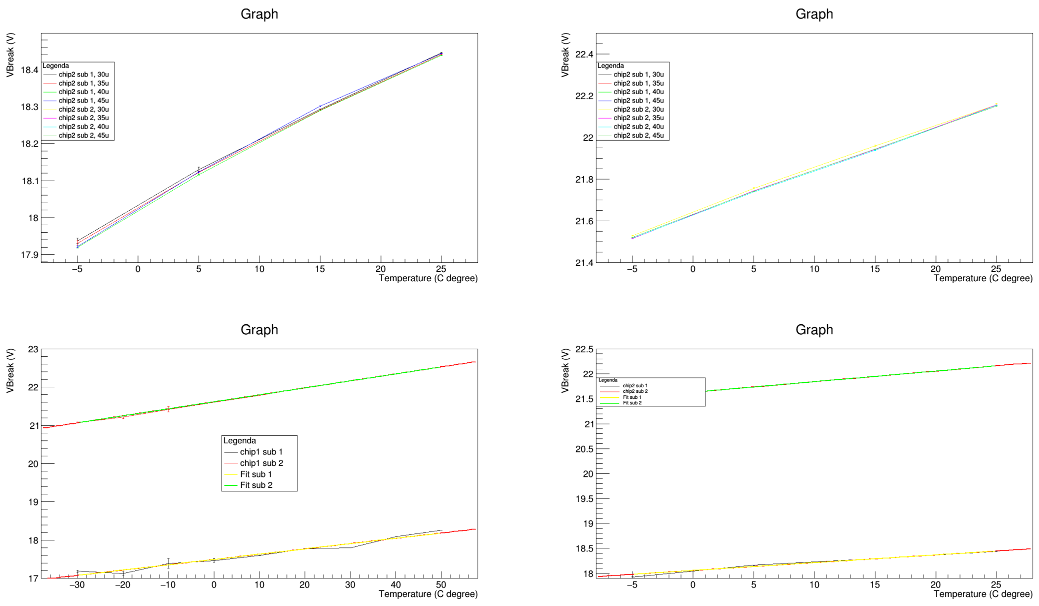


Figure 3.7: V_{bd} vs temperature graphs are shown. in the two graph on the upper part we note the independence from active area.

Chip	Subunity	$V_{bd,0}$ (V)	V_{temp} (V/K)
APiXC1	1	18.058 ± 0.002	0.0156 ± 0.0001
APiXC1	2	21.631 ± 0.001	0.02121 ± 0.00004
APiXC2	1	17.494 ± 0.028	0.014 ± 0.001
APiXC2	2	21.617 ± 0.002	0.0183 ± 0.0001

Table 3.4: Fit parameter of the V_{bd} dependence form temperature.

3.2 Optical Noise

One of the main issues of SPAD arrays is optical noise. In our measures there is the evidence of the cross-talk, thanks to the two mode of enable-reading processes, that are described in previous chapter.

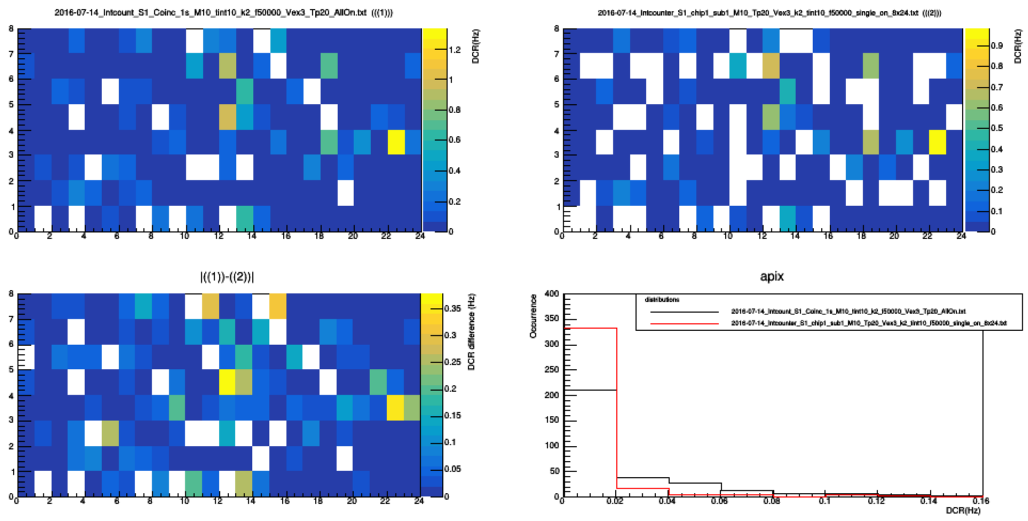


Figure 3.8: Scan of the same APiXC1 chip: in ((1)) is used the all-on algorithm and in ((2)) the single-on one. In the third scan, named $|((1))-((2))|$ the absolute difference of two previous scan maps. The fourth graph represent the frequency occurrences of the single-on scan (red) and all-on scan (black).

In the figure 3.8 is shown the application of two different enabling-reading operation to the same chip in stable conditions [1]: for the first one ((1)) all the matrix is turned on, then the content of the internal FIFO was read (call *all-on algorithm*); in ((2)) pixels are turned on one-by-one and an external counter saves counts of this pixel (call *single-on algorithm*). Both of that algorithms are reported in Appendix A. We note that there is a significant different between mean dark count rate between two operation. In fact there is a difference of the 30%[2].
The possible hypothesis is that there is a low percentage of new randomize noise in the difference, but the highest part of the difference is generate by the *communication* between pixels of the same array, called cross-talk.

Some simulations are done by Rech et al. imposing that the cross-talk event detection is due to reflection of photons on the lower silicon-air surface[16]. The emitters are represented by cylindrical emitting volume, that propagates his wave front, and then the light is partially absorbed. The cross-talk simulation for APiX are not treated in this work.

The main parameter that characterize the cross-talk is the **cross-talk coefficient** K, related to the coincidence counts. Coincidence counts are related by a composed probability of detection of both pixels. If C_1 and C_2 are dark count rate of two pixels and $C_{1,2}$ the coincidence rate:

$$C_{1,2} = 2C_1C_2\Delta t + K(C_1 + C_2); \tag{3.6}$$

where Δt is the coincidence time window, practically the signal width. So, we can measure the cross- talk coefficient as:

$$K = \frac{C_{1,2} - 2C_1C_2\Delta t}{C_1 + C_2}. \tag{3.7}$$

In this section there is the exposition of the evaluation of the horizontal cross-talk coefficient, and a brief analysis of vertical cross-talk between two layers.

[1]Conditions: temperature of 28 ^oC and $V_{SPAD} = 20.37V$. The cross-talk measurements are done only for the first subunity of the array, where SPADs are characterized by p+/nwell junction.

[2]In the data analysis and representation the cells affected by RTS signal are eliminated using the criteria that are eliminated data higher of the (mean $+ 3\sigma$) calculated after three acquisitions.

3.2.1 Horizontal cross-talk on APiXC1

To permit the evaluation of the horizontal cross-talk, only APiXC1 array has the coincidence out, that connected two rows outputs thanks to a couple of multiplexers, shown in figure 3.9. The output is connected to an external counter.

This settings permits to select a row that has a pixel with high DCR (for example with a RTS

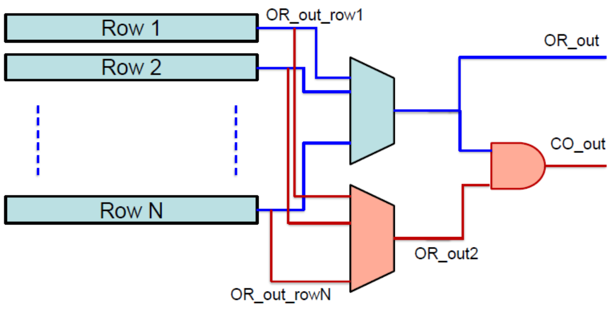


Figure 3.9: Intuitive scheme of the implementation of cross-talk coefficient measurement circuitry.

signal) as emitter and use the other pixels of other rows as detectors. Pixels of the same row of the emitter are not considered, because the external counter reads total counts of a row, so the cross-talk can't be measured. The measures are done for the first sub-unity at $V_{EX} = 3V$ at 20 and 0 ^oC for the first sub-unit. An analysis at 25^oC is done for second sub-unit.

The scans are manipulated with two filter conditions. The first one consist to impose an upper limit to detector pixel's DCR (10^4). The second one consist in an upper limit to detector-coincidence ratio (10). Denied pixels are changed with the nearest neighbor values.

Results are shows in figures 3.10, 3.11, 3.12, 3.13, 3.14, 3.15. In figures 3.10, 3.12 and 3.14

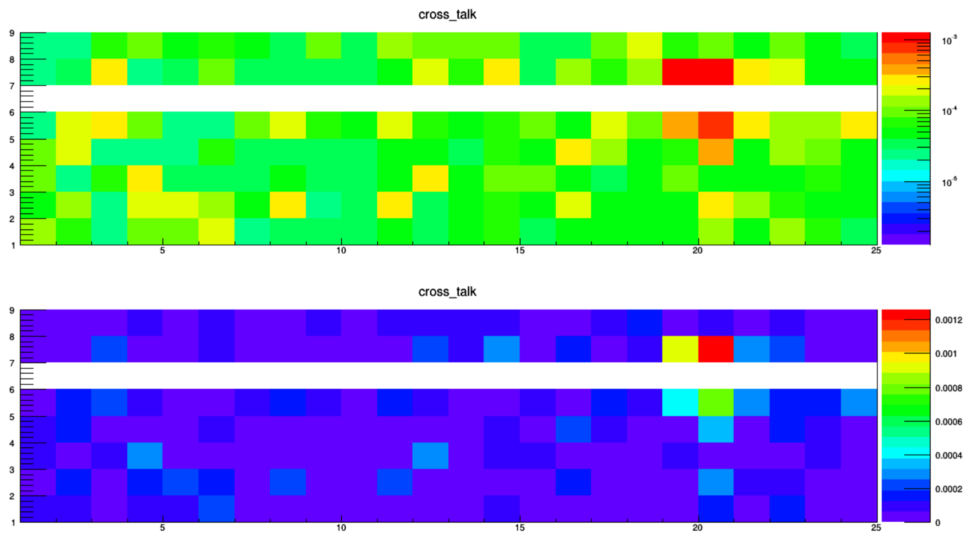


Figure 3.10: Heat-maps of the cross-talk coefficient, relate to cell (6,20) as emitter, at 0°C. Results are represented in linear scale (down) and logarithmic scale (up).

we can note that there is a bulk of high K values (near the emitter cell), and then a constant value of that, characterized by the same cell coloring. This variation of K in relation of the distance between the detector end the emitter pixels is underline in figures 3.11, 3.13 and 3.15. In this graphs the oscillation of values is due to the random presence of the noise, that gives an overestimation of the factor. We fit the data with an exponential function, that gives, when $T = 0^oC$:

$$K(x) = -9.455(103)e^{-(991.8\pm252.3)x}. \tag{3.8}$$

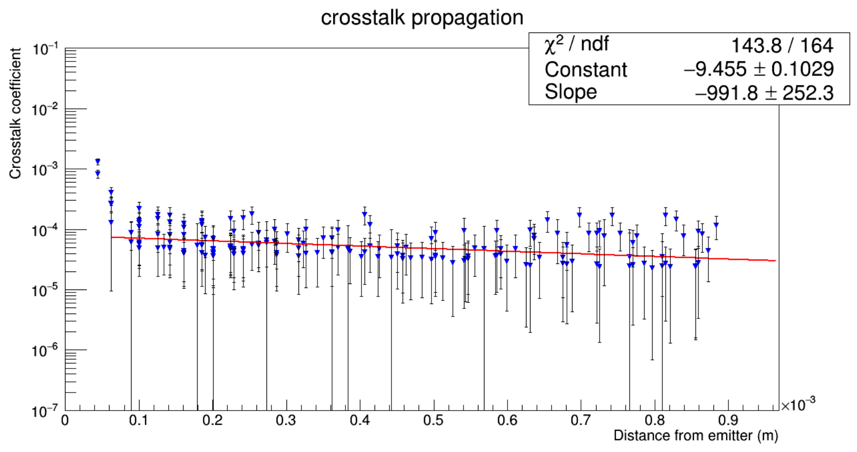


Figure 3.11: Cross-talk coefficient versus distance from the emitter pixel at 0 ^oC. The red fit is obtained by exponential interpolation.

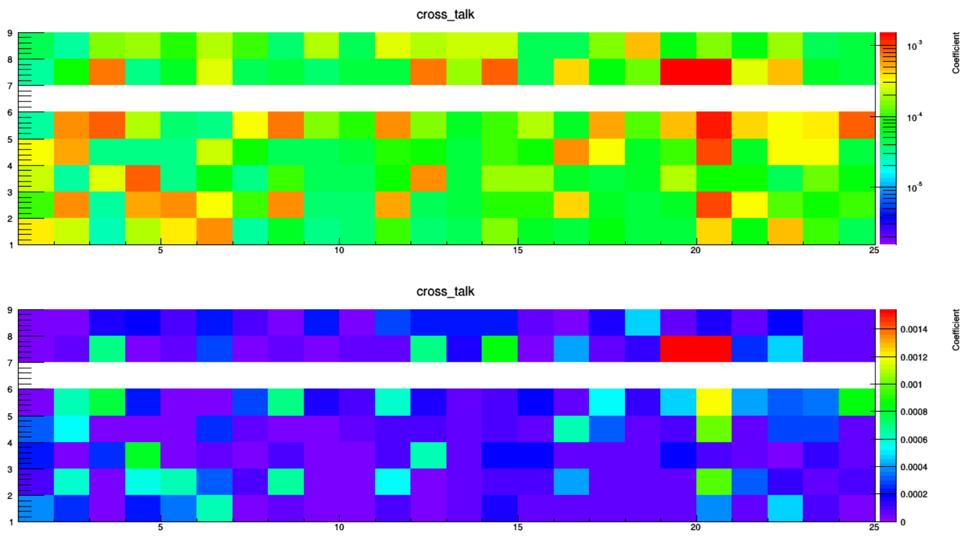


Figure 3.12: Heat-maps of the cross-talk coefficient, relate to cell (6,20) as emitter, at 25°C. Results are represented in linear scale (down) and logarithmic scale (up).

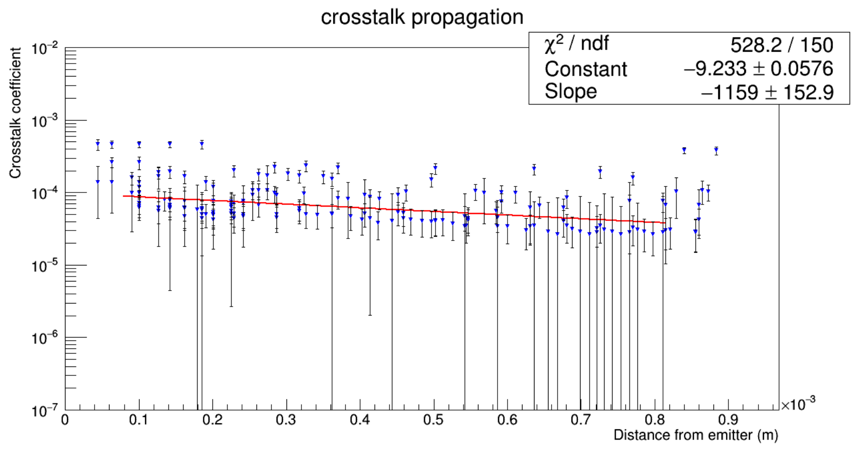


Figure 3.13: Cross-talk coefficient versus distance from the emitter pixel at 25 ^oC. The red fit is obtained by exponential interpolation.

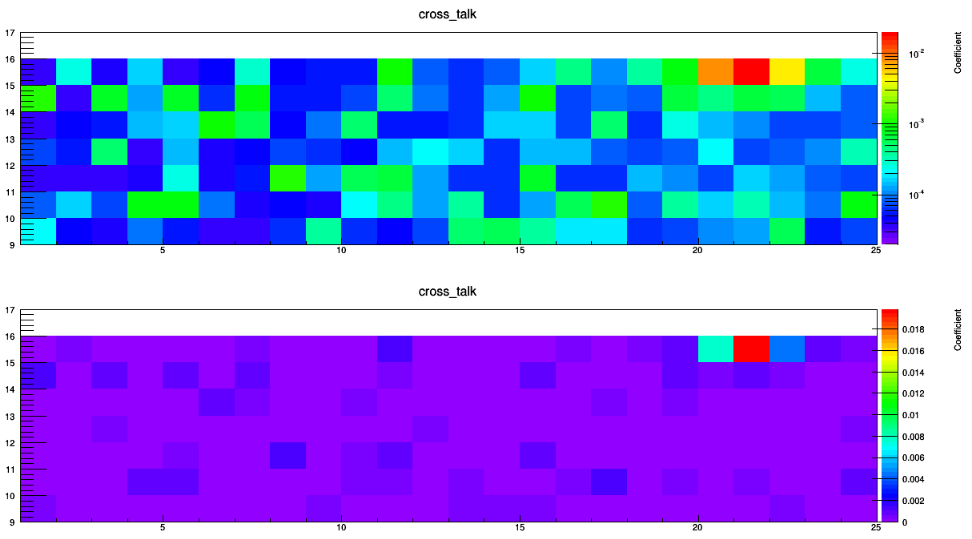


Figure 3.14: Heat-maps of the cross-talk coefficient, relate to cell (15-21) as emitter, at 25°C. Results are represented in linear scale (down) and logarithmic scale (up)0 To evaluate the data in a better way, a RTS filter was applied.

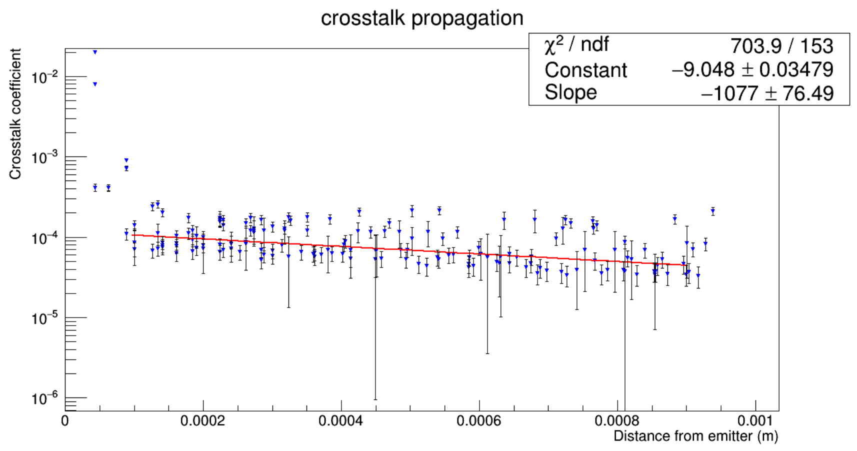


Figure 3.15: Cross-talk coefficient versus distance from the emitter pixel at 25 ^oC of the second sub-unit. The red fit is obtained by exponential interpolation.

And for $T = 20^oC$

$$K(x) = -9.233(58)e^{-(1159\pm152.9)x} \tag{3.9}$$

for the first sub-unit, and

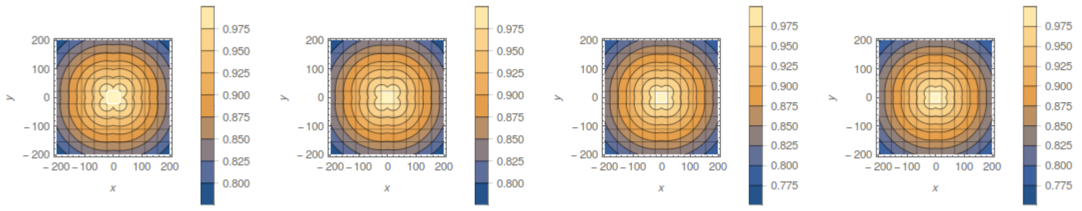
$$K(x) = -9.048e^{-(1077\pm76.49)x} \tag{3.10}$$

for the second one.

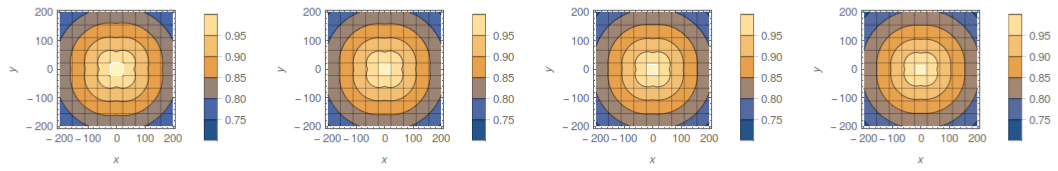
This relation can be improved with other measure at different temperature, but is not treated on this work.

We note that the cross talk coefficient at higher temperature is not only higher, but the influence from emitter pixel to detectors is bigger.In fact the interaction length is $1008 \pm 256\mu m$ at 0^oC, $862.8 \pm 113.8\mu m$ at 20^oC and $928.5 \pm 6.6\mu m$ for the second sub-unit. This results can be interpreted like that the effective active area in increased, due to the transverse permeability of the incident photons, like figure 3.16 for example. In this figure, the active profile is created using the same exponential decay from the borders of active areas.

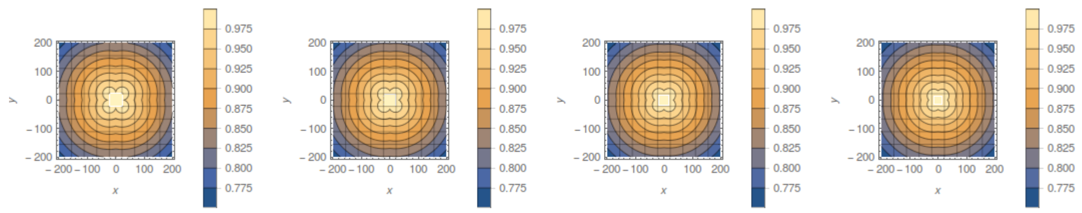
This data are must be compared with simulation similar of that Rech's one[16], but we can note that the cross-talk is contained by the temperature, like other noise features.



(a) Sub 1; 0°C



(b) Sub 1; 25°C.



(c) Sub 2; 25°C

Figure 3.16: Real active area for a single pixel of all sizes, due to the cross-talk. This type of profiles is a consequence of exponential decay of light attenuation. The analyzed area is underlined by white borders. Coordinates are expressed in μm.

The last consideration is that the hypothesis where the 30 % of total DCR events is acceptable. This fraction is usefully to give a qualitative evaluation of vertical cross-talk.

3.2.2 Vertical Cross-talk

We want to give a qualitative answer to this questions: due to the vertical coupling of two SPAD arrays, there is the possibility of the vertical crosstalk? The metal shielding gives an opportune optical isolation? To evaluate this we have select six symmetrical columns of the first sub-unity of the array, three shielded and three unshielded, with $43 \times 45 \mu m^2$ area. We have measured the DCR with *all-on* and *single-on* algorithms, and we analyze the differences, obtaining heatmaps in figures 3.17 and 3.18. We can try to give this evaluation, referring figure 3.17: the mean DCR is near $1.837 Hz$. The integration time for this scans is $1000 sec$, so we have 1837 counts. If the 30 % of this events is given by horizontal cross-talk (551 counts), vertical mean difference of coincidence will be 854, then 303 counts are given by vertical cross-talk or random coincidence, the 16.5%. Similarly for shielded pixels, only the 4.5% of total coincidences may are given by vertical cross-talk or/and vertical random coincidences.

We can hypothesize that the shielding works correctly and our evaluation gives an overestimation of the vertical cross-talk event percentage. Some of cross-talk events and DCR ones are gave by After-pulse and retarded photo-luminescence, given by an internal delayed recombination.

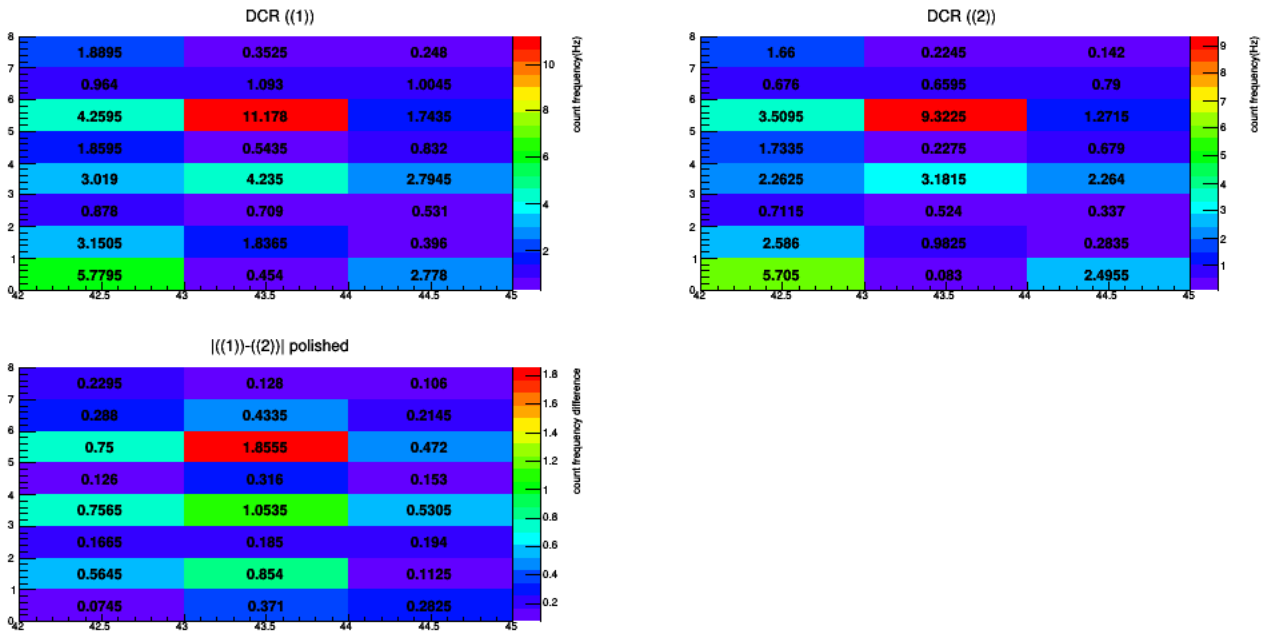


Figure 3.17: Vertical cross-talk evaluation for unshielded pixels, ((1)) is the scan with all-on algorithm, ((2)) is the scan with single-on algorithm.

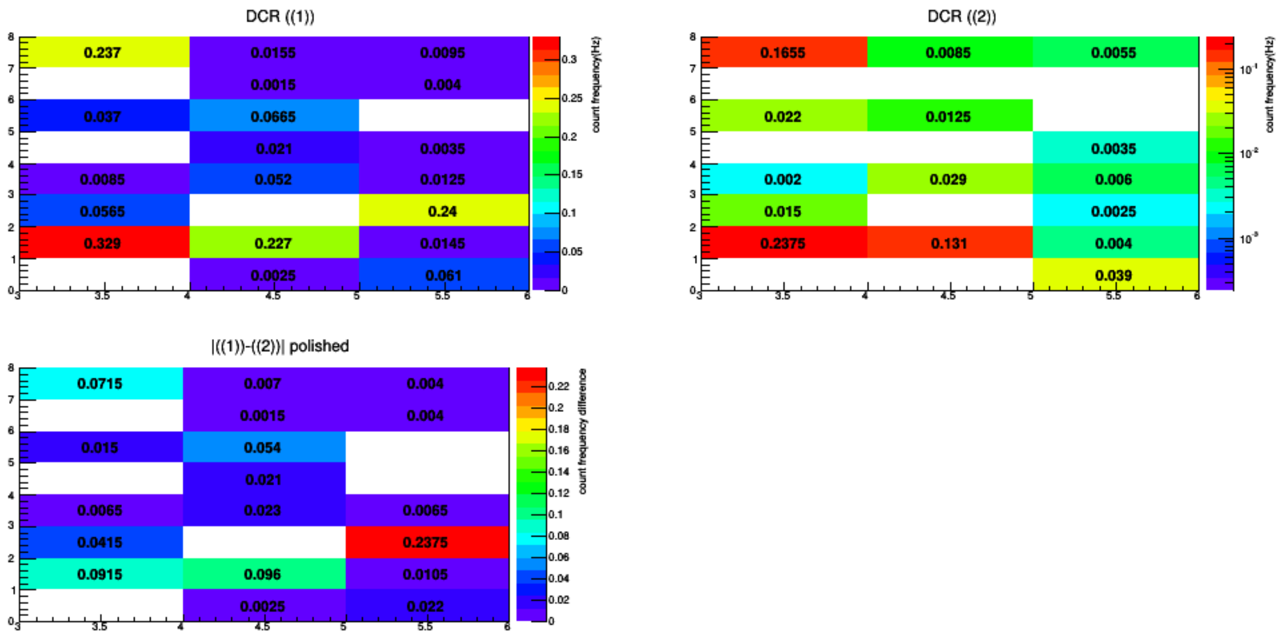


Figure 3.18: Vertical cross-talk evaluation for shielded pixels, ((1)) is the scan with all-on algorithm, ((2)) is the scan with single-on algorithm.

Chapter 4

Quantum efficiency and PDE.

In this chapter we analyze one of the most important aspect of the characterization of an APD or a SPAD: the quantum efficiency of photon detection.

The SPAD's efficiency, or the probability of detection of a photon with a fixed wavelength, is called Photo-detection Efficiency (PDE). PDE is the product of Quantum efficiency QE, the avalanche trigger probability $\epsilon_{trigger}$ and the fill factor FF:

$$PDE = QE \times \epsilon_{trigger} \times FF. \tag{4.1}$$

We define Quantum Efficiency the probability to create a e-hole couple after the photon impact. We can give also this operative definition: the QE is the ratio between the number of incident photons and the output current. It is dependent by temperature, wavelength and over-voltage applied. Mainly, it is related to the upper surface characteristics. Due to an energy gap of 1.2 eV for silicon, photons with a wavelength higher than 1020 nm are not detected, because they can't generate excitation in the lattice. In the chapter 1 there is described the probability of generation of the avalanche by a photon with impact ionization process.

The trigger probability ($\epsilon_{trigger}$) is the probability of an e-hole couple generate a new avalanche. It is function of the depth, the over-voltage, and is different of the two carriers. In fact, since electrons has n higher mobility than holes at fixed electric field, the trigger probability is higher for electrons. This probability depends on the depth, like every process described in the first chapter.

The fill factor (FF) is a geometrical data gives by the rate between the active ares and it's total area. It also is influenced by the area reduction and the separation between cells, occupied by quenching resistors [23].

4.1 Internal efficiency

For this characterization, we use a ultraviolet light source.
The ultraviolet region of electromagnetic spectrum is divided by near ultra violet (NUV, 400-300 nm), medium ultraviolet (MUV, 300-200 nm), far ultraviolet (FUV, 200-100 nm) and extreme far ultraviolet (EUV, 300-200 nm). Sometimes, the region between 200 and 10 nm, is called vacuum ultraviolet (VUV).

The main constituents of the atmosphere absorbs the UV light: MUVs are absorbed by the ozone, meanwhile the molecular oxygen absorbs FUVs. This aspect gives some problems to the observation on this spectrum, due to the absence of adapt devices for this usage. Although, since XXIX century, new optical technologies permits this kind of research.

The application of UV characterization is really important for particle physics. For example, dark matter detection and the description of double-β decays requests FUV's detectors.

The APD structure is most important for detection capability. For a *n-p* junction is optimized for visible spectra and not for UVs. In fact visible photons has an higher probability to trigger an avalanche, with a deepest conversion, than the UV light. This process was described in details in chapter 1.

Although, *p-n* configuration is the optimized option for UV,detection. In fact, electrons generated on the surface by UV photons are derived on the positive electrode, connected to the n-substrate, through the passage in a high-field region. This generates the avalanche. Visible electrons converts in deepest zones, and then they not pass into the multiplication region. For this reason, the efficiency is higher for UV photons.

There are some problems for UV characterization. Principally:

- the presence of a *epoxy* layer on the surface
- the presence of oxides layers, for example STI.
- conversion depth of ultraviolet photons in silicon.

The *epoxy* is an epoxy resin the can be applied on a device to protect bondings. This resin is opaque to wavelength lower than 300 nm. For this reason, unshielded APDs or SPADs can be utilized.

Some devices presents a protective layer against oxidation, made by silicon oxides (SiO_2), which absorbs UV photons under 150 mn. In the end an other protective layer is made by silicon nitrate, used like UV absorber [23].

4.1.1 Setup

The setup consist simply by a UV-laser pulser, characterized by a 200 kHz frequency, 200 ps of pulse duration and 350nm (UV) wavelength.

The measure consist of the evaluation of the ratio between detected triggers by the APiXC2 sensor and total triggers, measured using Teledyne oscilloscope. This measure is done for two cells, one shielded and one unshielded.

We note that the mean efficiency of unshielded pixels is $99.08 \pm 0.01\%$ and $1.35 \pm 10^{-5}\%$ for shielded one. For this reason we can say that the metal layer, use to shield the pixel, has a little light permeability, to consider in case of particle application.

The second analysis is the absolute efficiency profile for an unshielded pixel, using the same method mentioned below. We have tried to do this measure two times, in the first one the sensor is uncovered and in the second a $2\mu m$ Mylar layer is used as a cover. The profile is done by the application of a DC-sweep and the coincidence of trigger was measured.

The result has the same behavior has the DCR sweep, mentioned in the previous chapter(figure 4.1). In fact, the data are fitted using the error function[1]:

$$QE = QE_0 \times Erf\left(\frac{V_{SPAD} - V_{ON}}{\sigma}\right) \tag{4.2}$$

As a confirm, the V_{ON} fitted for both cases are compatible with the breakdown voltage of the first section of the matrix. The last part of both curves presents a decrease, because the DCR at high voltages increases, and than the probability of avalanche trigger decreases.

However, to have an absolute value of the Quantum Efficiency we have to sent at detector few photons per pulse. In this way the detector works in **single photon mode**.

To find the right configuration, the system in figure 4.2 was mounted. This system consist with the ultraviolet laser, where a lens for optical devices is applied to enlarge the light dot with 2.5 cm radius. In the other side of the guide there is the optical device, connected to a powermeter. The optics are covered using a metal disk with a hole oh 0.15 mm radius.

[1]The error function is defined as: $Erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

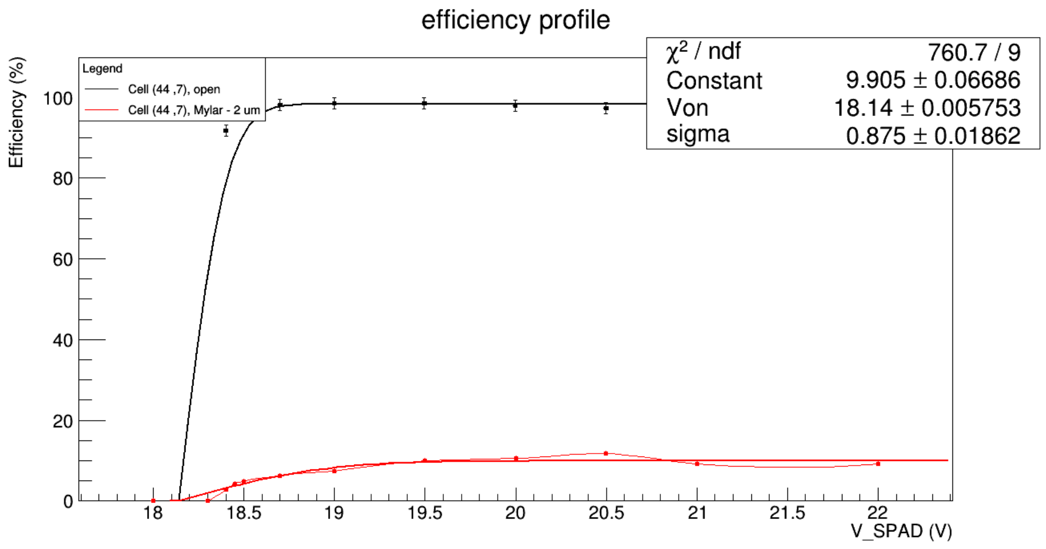


Figure 4.1: First attempt of measuring quantum efficiency using an UV laser.

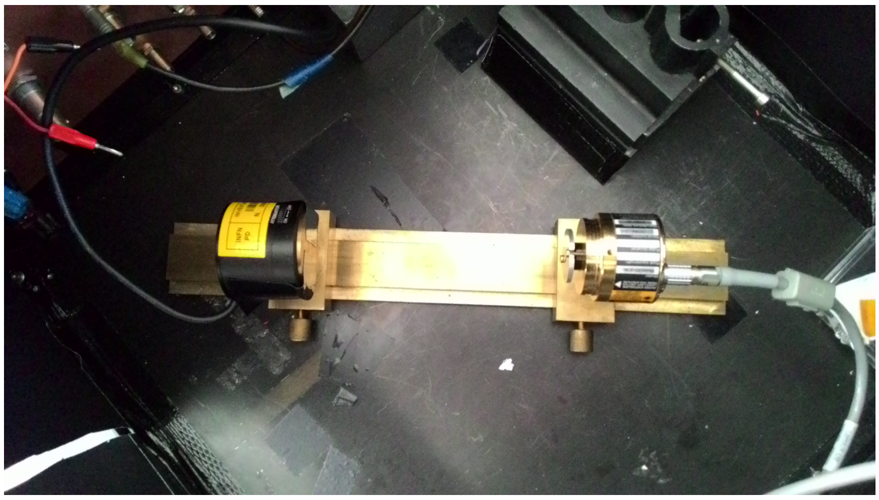


Figure 4.2: System used for the source calibration. The laser point was enlarged using a lens of a optical devices. The optical detector was masked with a 0.15 mm radius collimator (metal disk).

The calibration is obtained by the measuring of the light intensity in relation of the distance between source and the optical device. Indirectly we can obtain the number of photon collected by the optical sensor, using the Plank's relation:

$$P = \frac{dE}{dt} = \frac{nhc}{\lambda t_{pulse}} \rightarrow \tag{4.3}$$

$$n = \frac{P\lambda t_{pulse}}{hc} \tag{4.4}$$

where $\lambda = 350nm$ is the wavelength, $t_{pulse} = 200ps$ is the duration of a single laser pulse. The results are showned in figure 4.3, represented by green dots.

4.1.2 Results

For quantum efficiency measurement a special support, thanks to a 3D-printer. The support is shown in figure 4.4. This support and the laser source are attached to a linear guide, positioned into a dark box. To have a uniform diffusion on the detector, the source is covered with a Teflon foil.

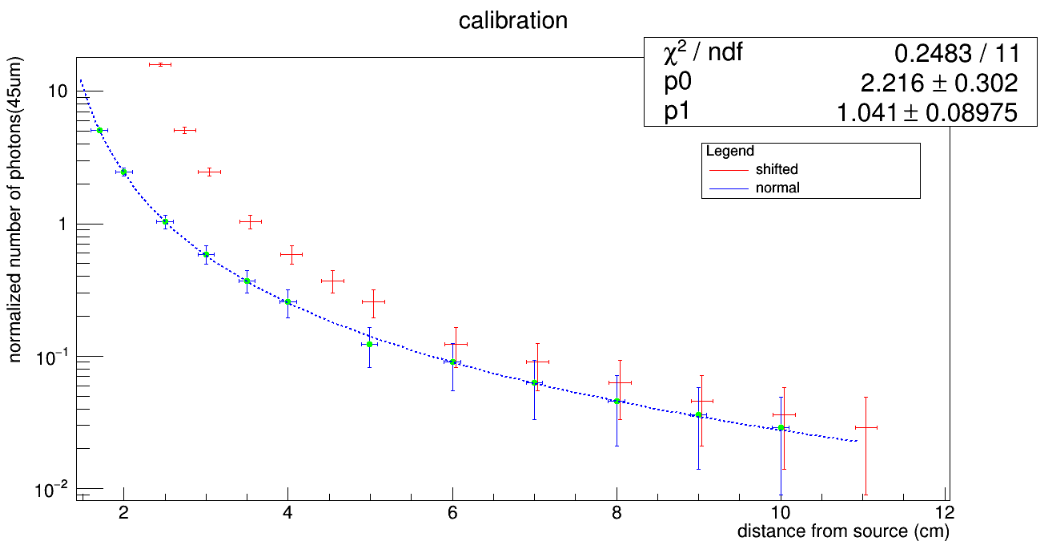


Figure 4.3: Calibration profile of the laser source. The figure also shoes the response of the alignment algorithm (red dots) compared to original data, normalized for the total area of one pixel($75 \times 50 \mu m^2$). The horizontal shift is obtained by the fit (blue dotted line).

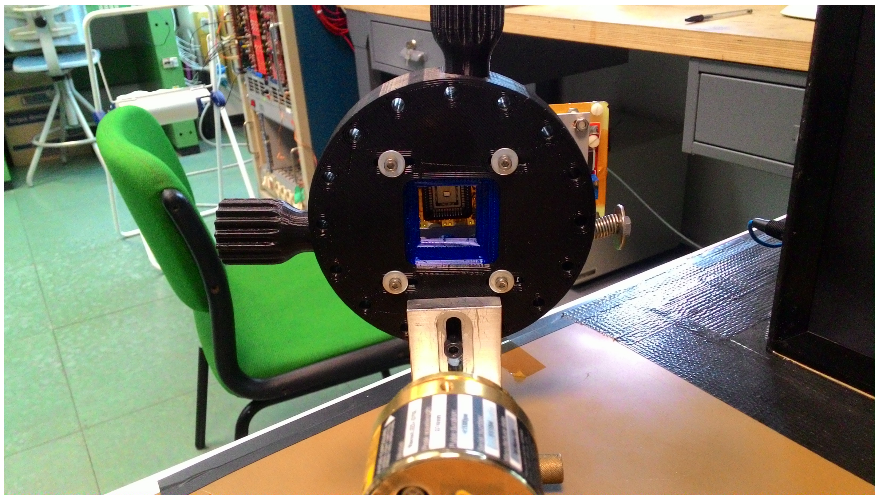


Figure 4.4: Support and sensor created for APiXC2.

Counted photons at different distances are measured, and then compared to the same result, obtained using a calibrated diode. Each result fit is proportional to x^{-2}, but arbitrary frames must be normalized.
To align the rest frame of the calibration and SPAD measurement, the detected photon of each session are fitted with:

$$f(x) = k\frac{1}{(x - x_0)^2}, \tag{4.5}$$

then, the X-axis' values of the three measures are aligned in function of the obtained parameter x_0:

$$\{X_{Values}\} \rightarrow \{X_{Values} + x_0\}. \tag{4.6}$$

This method is applied also for the calibrated diode data. The result of the fit (Blue dotted line) and the consequent data shifting (red dots) are show in figure 4.3 as an example. Then the trigger percentage detected for the pixel (44,7) from the two subunits of the array are divided with the correspondent value of the calibration curve to obtain the Fill Factor profile of the pixel, shown in figure 4.5.

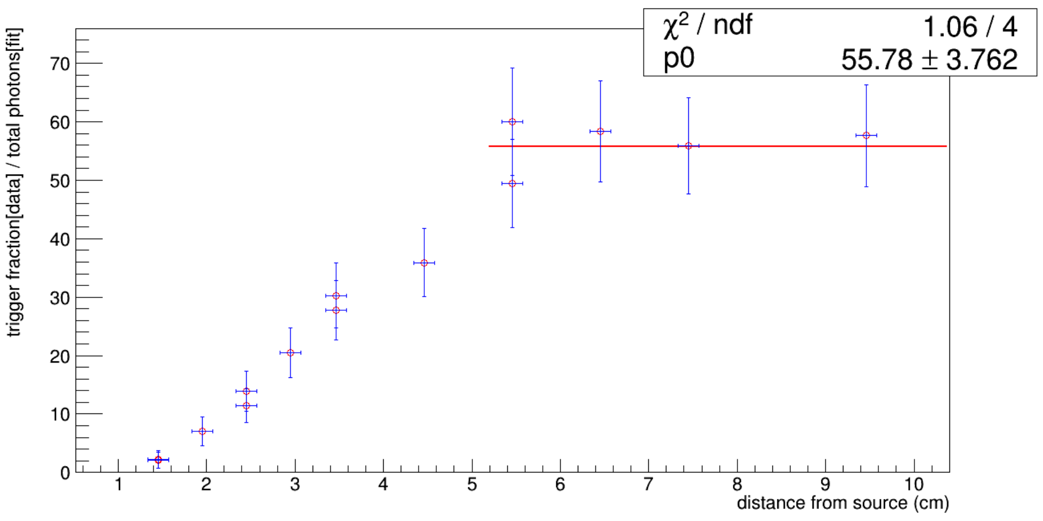


Figure 4.5: Fill Factor profile of pixel (44,15).

From figure 4.5 we can note that on the single photon region ($x > 5cm$) the measured fill factor saturated at $(55.78 \pm 3.762)\%$. The expected Fill Factor can be calculated through the definition:

$$FF = \frac{active\ area}{total\ area}. \tag{4.7}$$

Referring to the figure 4.6:

$$FF = \frac{45 \times 43\mu m^2}{75 \times 50\mu m^2} = 52\%. \tag{4.8}$$

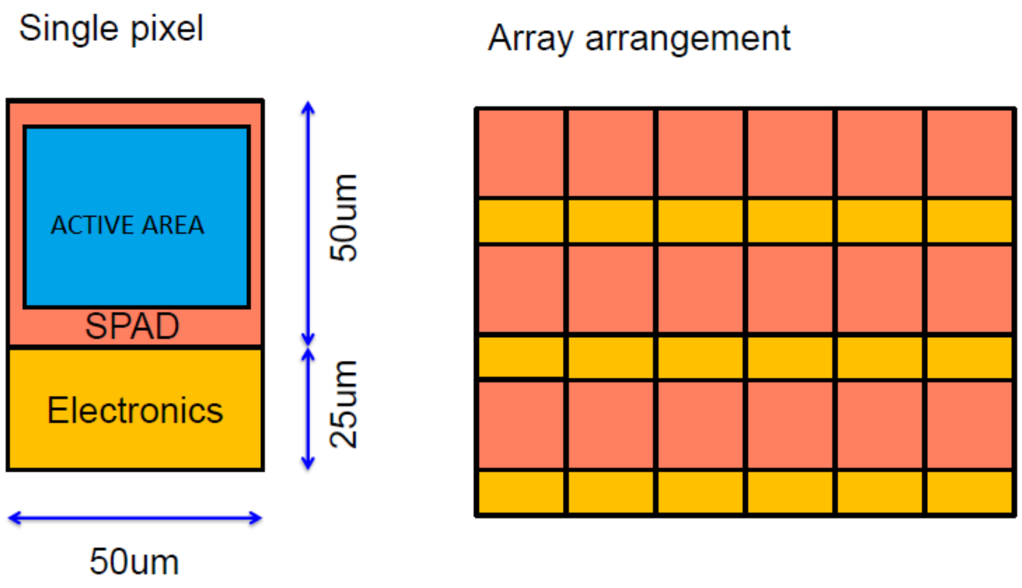


Figure 4.6: Pixel and array areas arrangement.

We can conclude that the experimental result are compatible with the expected one. The difference between the two values is maybe related to the non ideal contention of the avalanche.

4.2 External efficiency

To measure the relative efficiency we have to build an apparatus different than that used before.

The setup is made by a reticular monochromator JOBIN YVON HR 250, which split the polychromatic light beam originated by light source in its monochromatic components. Thanks to a step-motor an output light beam with a fixed wavelength was selected. This beam is sent by a MgF_2 lens[2] to a magnesium fluoride, positioned at 45° respect the beam direction. This window has a role of a splitter. It splits the beam in two new ones: the reflected one (n=0.08) e the transmitted one. The transmitted beam is analyzed by a PMT or the APiX sensor alternatively. The reflected beam is received by an other PMT, which normalize the light intensity and the variation of this quantity during the measuring session.

Both reflected and transmitted beams walk through two collimators, which they are positioned in front of detectors. The collimator positioned along the transmitted beam's direction ha s $3mm$ diameter and, over the collimator, a filter can be positioned. This filter is used to deleting some wavelength regions. The set up schematic is presented in figure 4.7.

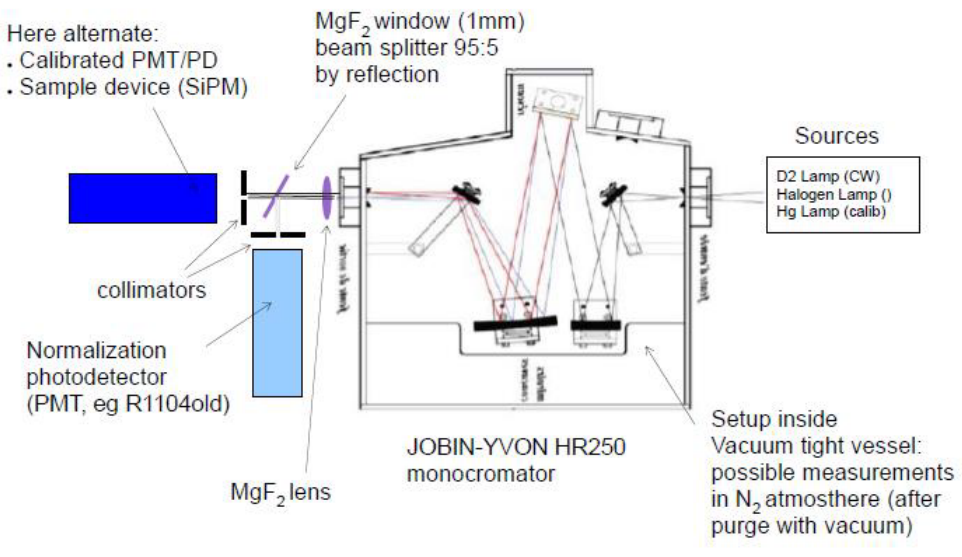


Figure 4.7: Apparatus Schematics.

4.2.1 Sources

During this session many radiation sources are used to study different spectral range. The sources are: Deuterium lamp with MgF_2 window, which works in 180-600 nm range and an halogen lamp with 250-950 nm range. Both lamps can be present intensity changes during the measurement cycle. for this reason a normalizer PMT was used. The halogen lams works through the heating of a filament, generating a black-body emission. The deuterium lamp works thanks internal discharge process. The spectrum of a D_2 lamp is related to the electrons' scatterings and excitation.

To calibrate the apparatus a discharge lamp in $HgCd$, with a particular spectrum, related to its electronic levels.

[2]MgF_2 lens are used because they are transparent in FUV region.

4.2.2 Monochromator

The monochromator is the core of the apparatus. It present a input slit, which can be set. In this slit the beam comes into the monochromator. The beam is reflected thanks to a plane mirror en send to a spherical on. The focus of the spherical mirror coincides to the position of the collimator, than the mirror project the beam on the lattice. The used lattice -*Blazed Holographic 530 25*- has 1200 rows/mm of density and works in reflection. The diffracted light at first order is send to a concave mirror, which focus the beam to the output slit, after an other reflection through a plane mirror. Thanks to a step-rotor the wavelength can be set through the lattice rotation. The scanning is controlled using a electronic controller. The terminal send information to this component and than to the motor driver (a rotary encoder). The real-time measure of the selected wavelength is made by a calibration. In figure 4.8 the monochromator was shown.

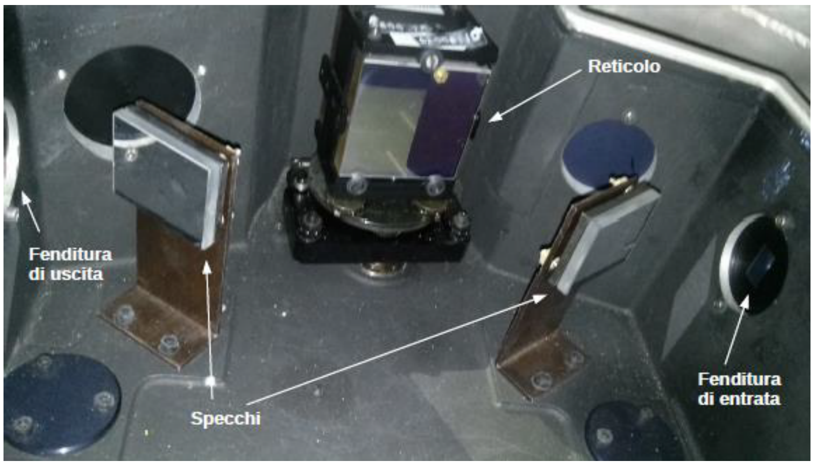


Figure 4.8: Interna structure of JOBIN-YVON HR250 monochromator. The spherical mirrors are not included.

The mirrors are made in aluminium covered with magnesium fluoride. Optical components of UV cannot be made in glass, because absorbs UV lower than 320 nm. to work in regions below 180 nm the apparatus works in N_2 environment, because oxygen absorbs this wavelengths.

Let now focus on the lattices. Lattices generally operates in transmission. A lattice is made of a substrate of optical materials, where are applied a huge number of slots. The slotted surface is covered by reflecting material, like aluminium. The geometry and periodicity of the lattice are fundamental for the performances, so this device requires care.
The principal parameters of the lattice is the step, opposite to the slots density. In figure 4.9 is shown the interference mechanism of reflective lattice.

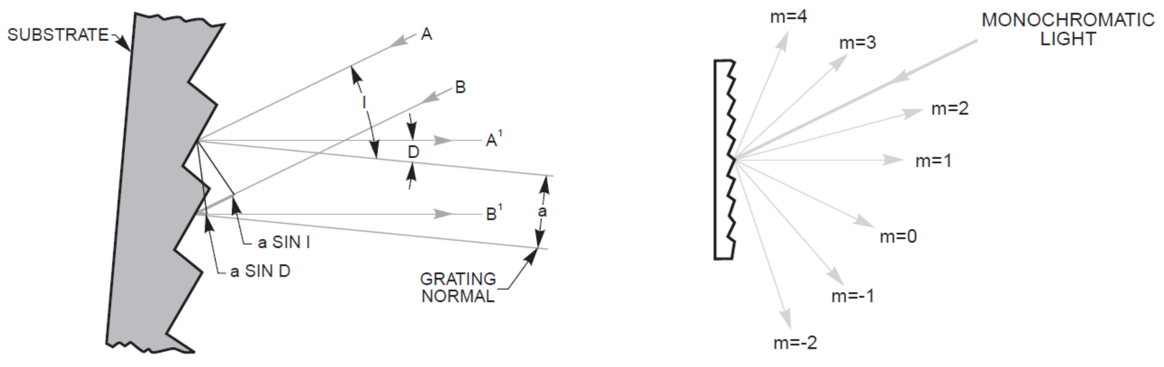


Figure 4.9: Lattice profile and interference mechanism.

Two parallel rays with wavelength λ engraves on the surface of the lattice with an angle I upon two adjacent slots, and then are reflected with an angle D [3]. Reflected rays interfere constructively if the optical path difference is multiple of their wavelength:

$$a \cdot (\sin I + \sin D) = m\lambda, \tag{4.9}$$

where a is the lattice step. The reflection of a parallel light beam by N steps is analogue with that one with two rays: the peaks becomes thick with the increasing of N, because the distance between a maximum and an adjacent minimum is λ/Na. The variation of m gives the different diffraction orders, with different angles. Because the incident direction is fixed by the setup geometry, the position of the lattice will be change within a rotation. For this reason is convenient to transform the last equation in a equivalent one:

$$2a \cdot \cos\phi \cdot \sin\theta = m\lambda, \tag{4.10}$$

where $\phi = \frac{D-I}{2}$ and $\phi = \frac{D+I}{2}$, so the angle necessary to see a 0-order diffraction. Consequently, the variation of θ scans different orders of diffraction. Although, referring to figure 4.10, at the same position there are present different orders, for λ, $\lambda/2$, $\lambda/3$, etc.. When θ is fixed, the position of the lattice and the geometry with ϕ, the product $m\lambda$ is constant. This factor gives a contribute to the error propagation, giving a systematical error to the PDE result. This problem can be solved using high-pass filters[23].

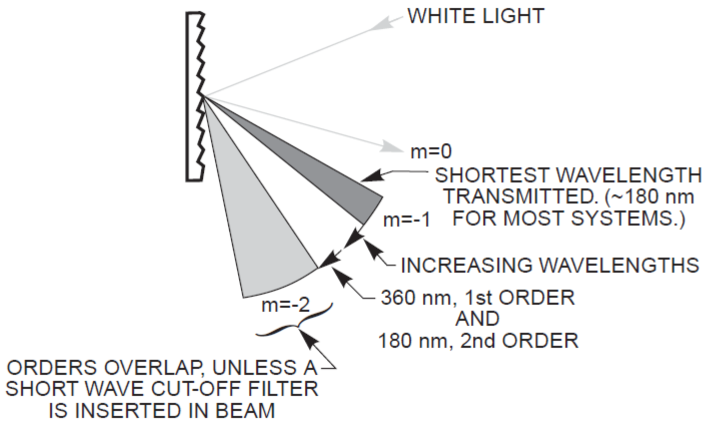

WHITE LIGHT
m=0
SHORTEST WAVELENGTH TRANSMITTED. (~180 nm FOR MOST SYSTEMS.)
m=-1
INCREASING WAVELENGTHS
360 nm, 1st ORDER AND 180 nm, 2nd ORDER
m=-2
ORDERS OVERLAP, UNLESS A SHORT WAVE CUT-OFF FILTER IS INSERTED IN BEAM

Figure 4.10: Superposition of different diffraction orders.

An other parameter of a lattice is the resolving power, given by $R = \lambda/\Delta\lambda$. R represent is the minimum distance where two different m-order peaks for wavelength $\lambda + \Delta\lambda$. In base of Rayleigh criteria, the maximums separation is $m\frac{\Delta\lambda}{a}$ and the distance between two adjacent maximum and minimum is $\frac{\lambda}{Na}$. Then:

$$R = \frac{\lambda}{\Delta\lambda} = mN = N\frac{2a\cos\phi\sin\theta}{\lambda} = w\frac{\cos\phi\sin\theta}{\lambda}, \tag{4.11}$$

where w is the beam thickness. Then, the resolving power depends on the wavelength, the amplitude and the geometry of the lattice. The resolution of the lattice used is 0.1 nm at 500 nm.

The reticular efficiency is defined as the rate between the intensity of output beam and the input one. This quantity, related to wavelength, is determined by the material, the shape, the width and the inclination of slots. The way to optimize the efficiency at fixed wavelength is called

[3] All angles a referred respect the normal direction of the lattice, with the convention that the anticlockwise direction is positive.

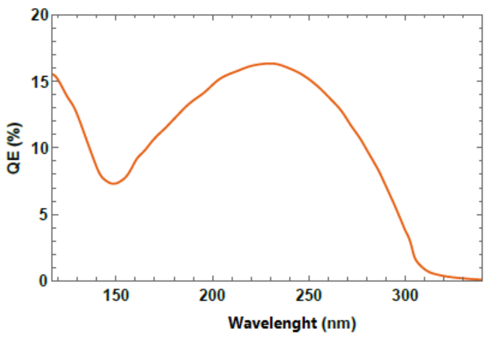
blazing, made by a geometry modification. Generally, the efficiency is considered at first order. At higher orders, the efficiency maintain the same functional form, with a scaling of wavelength given by m and a decrease of the efficiency too. The lattice used has a reticular efficiency with a maximum at a *blaze* 250 nm wavelength.

4.2.3 Calibration and Normalization using PMTs

During the experiments, diverse *Hamamatsu* vacuum PMTs are used for calibration. Because the efficiency of this devices is known, we can use the efficiency spectrum profile for the derivation of the PDE of the SPADs.
Each PMTs had an HV input, which is plugged in a preamplifier module. in tables 4.1 are reported the optimal working points, and in figures 4.11 and 4.12 are reported the external quantum efficiency profile. The gain of the PMTs is defined with an relative error of 10%.

Name	Range (nm)	HV (Volt)	Gain	Dark Current(nA)
R1080	117-340	1100	10^6	0.4
R7195	292-705	1800	10^7	5
R1104n	200-880	1000	$2 \cdot 10^7$	20
R1104o	200-920	1000	$2 \cdot 10^7$	10

Table 4.1: Normalization and Calibration PMTs.



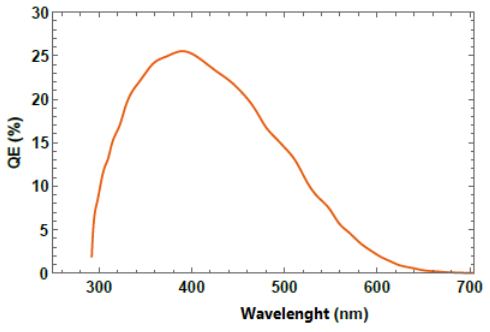
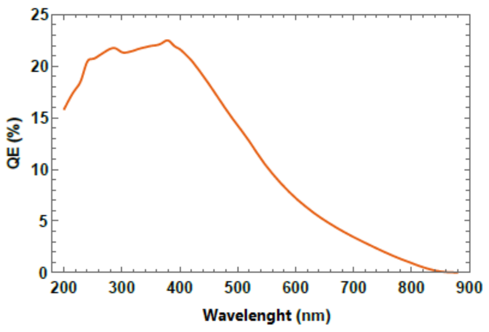


Figure 4.11: QE spectrum of R1080 (left) and R7195 (right)



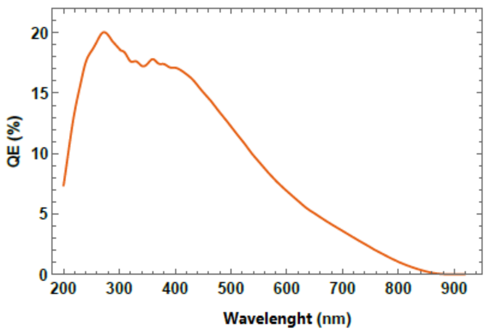


Figure 4.12: QE spectrum of R1104n (left) and R1104n (right)

To normalize spectra and consider the variation of intensity of the lamps during the experiments, for all measures R1104o is used, because a larger spectral range.

4.2.4 Optical Filters

During measurements, two optical filters are used to eliminate undesirable components of light, given by second orders maximums. That filters are simply quartz windows, which permits to work at the minimum of 180 nm. For the measures with the deuterium lamp an high-pass filter was used. The cut-off is 475 nm. The cut-off is chosen to cut the spectrum where the second order maximums detected with a wavelength scanning. In fact, The spectrum of deuterium lamp stops at 600nm, and 950 with the halogen lamp. In the table 4.2, filter specifics are shown.

Name	cut-off(nm)	width(nm)	Diameter (nm)
WG 305	305	2	8
GG 475	475	2	8

Table 4.2: Optical filters specifics.

4.2.5 DAQ chain

PMTs are powered by HV generator and the output signal is extracted by the last dinode, to reduce the noise. The output signal is connected to the amplifier thanks to a BNC cable.

APiX chips are connected to the DC supply, which is controlled by a GPIB-USB-B driver. The GPIB-USB-B is connected to a terminal, which set automatically the DC level, to guarantee the lowest DCR. The OUT-OR output of the chip si connected to a 12-channels counter, connected via LAN to the terminal.

The signal processed by PMTs are amplified and digitalized trough a ADC and then saved. We can set onto the terminal:

- The gain of the amplifier for the normalizer and even for the calibration PMT;
- The wavelength range (150-600 for deuterium lamp and 250-950 for the halogen lamp);
- The *wait* time between measures (default: 1s);
- The integration time of the input signal (5 μs for APiX and 1 s for the PMTs).

For each scan the rotor spins the lattice, the system is interrupter for *wait* time and the acquires the measure cycle, scanning the selected spectrum.

PMT's DAQ chain converts 1 nA in 1000 counts. The gain is $G = 1$. If $G = 2$ the ADC converts 1 nA in 100 counts.

4.2.6 Apparatus limits

The hypothetical limits for the apparatus are:

- the rotor may spin the lattice in an incorrect position, giving a systematic shift into the measure cycle;
- the light intensity is variable, then the normalizer PMT is must be used;
- monochromator can superpose second order maximums onto first order ones. For this reason the beam cant be absolutely mochromatic.
- The APiX chip must be positioned onto the beam, with a compatible area.

Some of this problems can be resolved with some tricks during measurements and data analysis, e.g. the elimination of second orders maximum using filters. The PDE is studied in function of the impact of those limits.

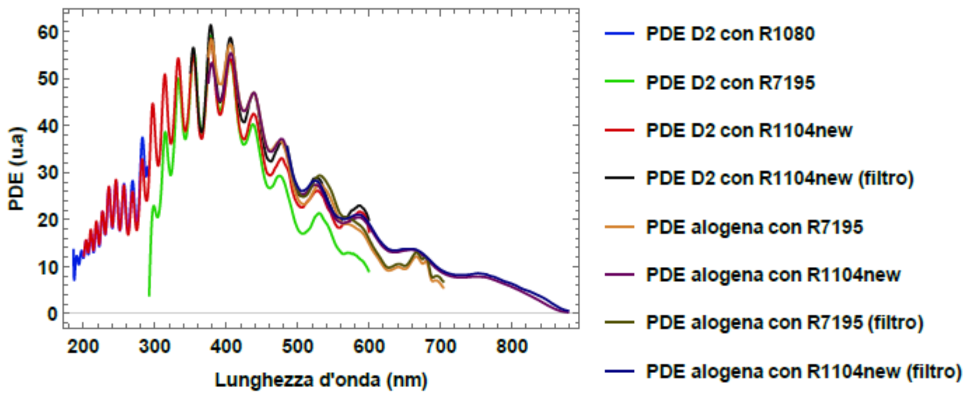


Figure 4.13: SPAD (44,15) PDE.

4.2.7 Results

In figure 4.13 are shown all the PDE measures for the SPAD (44,15), obtained using the halogen lamp and the deuterium one. The spad has a reverse excess bias of 2V. For the deuterium measures PDEs are measured using R1080 and R7195, without filters; and R1104n is used both with and without filters at 305 nm.
For the halogen lamp R7195 and R1104n (with 475 nm filter) are used.

The results are connected imposing that the match trough the regions without maximums of second order. where the PDE remains the same.

The comparison of PDEs shows that R1080 and R7195 using a Deuterium lamp are unattainable when their efficiency decrease. In fact, there is a strong correlation within the PDE and QE spectrum [23].

In the end, the result are consistent with the last IQE measurements at 380 nm and the FF expectation.

Chapter 5

Particle characterization.

In this last part we focus on the characterization of the chip using charged particles. This experiment was done into the laboratory of embedded electronics at University of Trento. The objective of this test is that to find a consistent counting rate, uncorrelated with the noise. This test becomes useful for the beam test at CERN[7].

5.1 Characterization with ^{90}Sr source.

For this characterization is used a ^{90}Sr source. This source decades in β^- mode with a branching ratio of 100 %, then this source produce electrons with a momentum of $545,9keV$. The lifetime of this source is 29 years, and after 24.498 days is in secular equilibrium with ^{90}Y [2]. The nominal activity of the source is 36 kBq, so the source *produces* 36000 electron, one for decay.

The source has a token-like shape, where bottom and lateral sides are shielded thanks lead deposition. For this reason, we can hypothesize that the spatial emission of electrons is contained in a 2π sr solid angle.

The setup consist in a little modification of that used for APiXC2 characterization, explained in chapter 2. The source is inserted on the base of the climatic chamber, in front of the unshielded surface there is the sensor, where the active area is put in front of the source. Of course the sensor is connected to the carrier in the same way shown in the chapter 2. A simplified scheme is shown in figure 5.1.

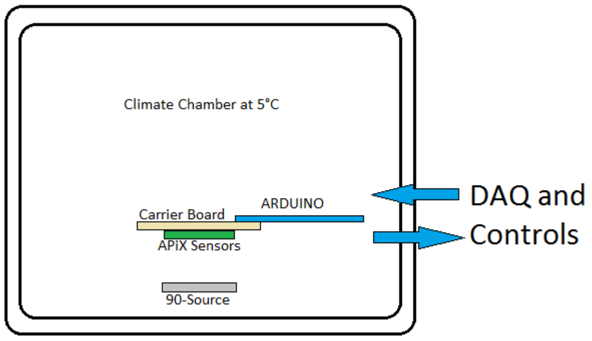


Figure 5.1: Simplified scheme of the experimental setup.

Due to the dependence of the noise to the environment temperature ,described in chapter 3, the temperature of the chamber was stabilized at 5 ^oC. In this way the noise is reduced.

First of all we have measured the DCR pattern before the particle characterization. Then the chip is exposed to radiation. We have done this measure two times, with two differents integration times: 1000 and 10000 sec.

Now let focus on the experimental results, then they will comparate with simulation results.

5.1.1 Experimental results

From both scans are created three frequency heat-maps: the exposed situation, the dark situation and the heat-map of absolute difference between two previous situations. These results are represented in figures 5.2 and 5.3.

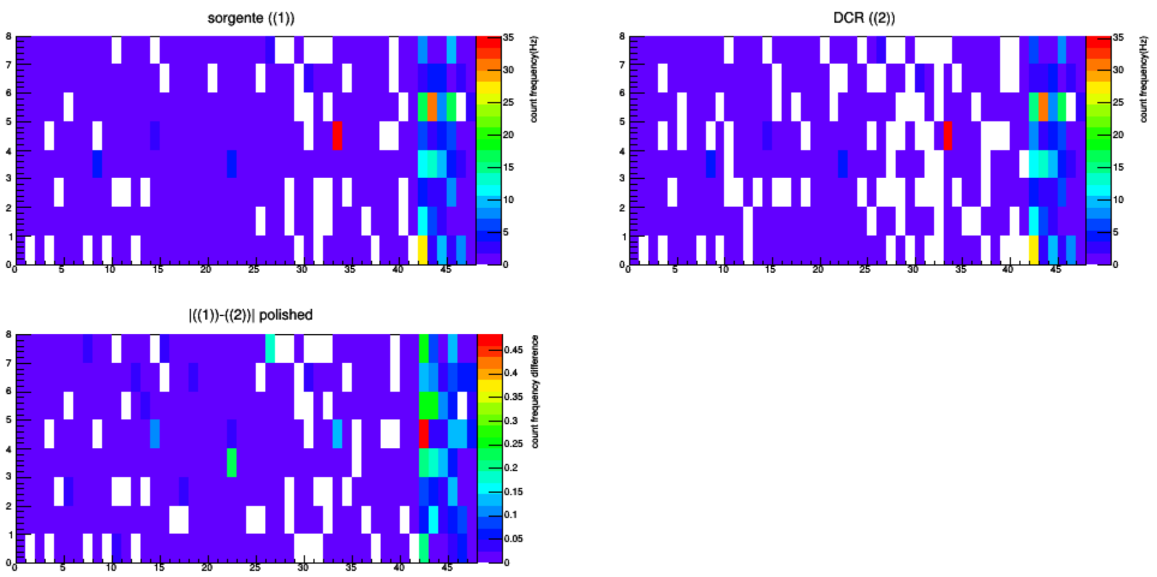


Figure 5.2: Heat-maps in 1000 seconds of integration: ((1)) is the map of exposed chip, ((2)) is the scan of DCR and the third map represent the absolute difference of previous maps, operated pixel by pixel. Some of the DCR map are normalized. In this way, the RTS is erased.

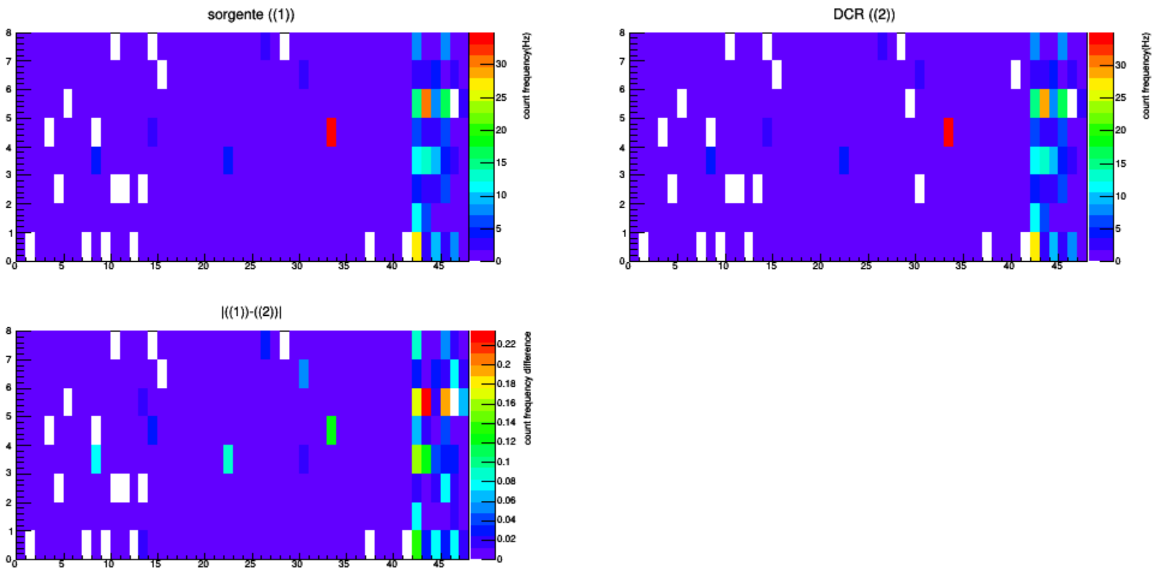


Figure 5.3: Heat-maps in 10000 seconds of integration: ((1)) is the map of exposed chip, ((2)) is the scan of DCR and the third map represent the absolute difference of previous maps, operated pixel by pixel.

We note that last six columns has higher count frequencies because their are not shielded. In fact, they have an higher DCR component. The difference pattern is constant in the rest of the array.

The only way to know that there is an consistent number of events is to analyze the average number of events.
We found that the mean number of events per pixel when the chip is exposed for 1000 seconds is:

$$N_{exp,1000} = 850 \pm 92, \tag{5.1}$$

and the mean value of electrons-generated events per pixel, that are calculated from the absolute difference, is:

$$N_{\beta,1000} = 17 \pm 13, \tag{5.2}$$

then there is a confidence level of 57.1% of acceptance. Of course the error of this result is huge, due to the low number of events (approximately seven thousands of total events). In fact the result changes with higher integration time. When every cell is integrated for 10000 seconds, total events per cell measured is:

$$N_{exp,10000} = 8001 \pm 89 \tag{5.3}$$

and electrons-generated events per cell are:

$$N_{\beta,10000} = 81 \pm 9, \tag{5.4}$$

with and acceptance of 81.8%.
Of course this results are must improved with simulation the spatial distribution of incident electrons, because the experimental data is under the threshold of 3σ for a certain detection.

The same experiment is simulated using PENELOPE (Appendix B). We obtain that, accounting for the fill factor, the expected value from simulation is a factor 1.2 larger than the experimental one. This result would imply an efficiency of 80%, but needs further investigation.

5.2 Test beam of APiX at CERN

In mid September 2016 two APIX chips were exposed to an high energy muon beam at CERN North Area. Tracks were measured also by an external silicon strip tracker with a resolution of $80\mu m$. Tracks were observed by both the chips APIX via coincidences. The two chips were aligned to better than $40\mu m$. Detection efficiency to minimum ionizing particles is being studied both using the information of one chip againt the other and using the external tracking data. Preliminary indications are that the efficiency tends to saturate to 100% at $\sim 2V$ overvoltage.

Conclusions

In this work we discussed some results related to the characterization of the prototype APiX chip. The main results follow:

- The breakdown and the noise background are related to the temperature with the expected dependence;
- The Dark Count Rate is highly suppressed due to the double-layer configuration;
- We have measured the crosstalk probability and the "cross-talk geometry", caracterized by a crosstalk lenght(800 μm);
- The internal quantum efficiency at 380 nm wavelength was measured. The data (for shallow p+/nwell junction) are consistent with a detection efficiency, which saturates at 100 %, at 2V of overvoltage and and active area of 52 %.
- A first characterization with ionizing particle was done. Results of β particles are not conclusive because the source activity was too low. Preliminary results of a test beam at CERN with Minimum Ionizing Particles (MIPs) indicate a very high efficiency ($\simeq$ 100% at 2V overvoltage) for the detection of charged particle at minimum of ionization.

The analysis of the prototype chips will be continued in depth during the forecoming months. By the way we can already conclude that the preliminary results are quite encouraging and allow to proceed with the next APIX project steps. In particular during Winter 2016 the development of a larger surface detector, with 5 mm^2 of active area ($\simeq 10^4$ pixels), will be carried on.

Appendix A

Algorithms and firmwares

A.1 APiXC2 Enabling firmware ARDUINO DUE (SAM3X8E ARM).

```
void enable (int col_sel, int row_sel)
   {
        int i,j;
        digitalWrite (TEST_N, LOW);
        //idle state register set
        digitalWrite(VSR_RST,LOW);
        digitalWrite(VSR_RST,HIGH);

        //start writing
        for (i = 0; i< 16; i++)
        {
                //shift register activation
                digitalWrite(VSR_CLK,HIGH);
                digitalWrite(VSR_CLK,LOW);
                //row register reset
                digitalWrite(EN_RST,LOW);
                digitalWrite(EN_RST,HIGH);
                for (j = 0; j < 48; j++)
                      {
                       if ((i == row_sel) & (j == col_sel))
                         {
                               digitalWrite(EN_IN,HIGH);
                               digitalWrite(EN_CLK, HIGH);
                               digitalWrite(EN_IN,LOW);
                               digitalWrite(EN_CLK, LOW);
                         }
                       else
                         {
                               digitalWrite(EN_CLK,HIGH);
                               digitalWrite(EN_CLK,LOW);
                         }
                      }
               }
        //resetting position
        digitalWrite(VSR_RST, LOW);
        digitalWrite(VSR_RST, HIGH);
        for (int i=0; i < (row_sel+1); i++)
           {
                digitalWrite(VSR_CLK, HIGH);
                digitalWrite(VSR_CLK, LOW);
           }
        // Enable register loading completed
        digitalWrite(TEST_N, LOW);
        delayMicroseconds(20);
        digitalWrite(TEST_N, HIGH);
        delay(2000);
   }
```

A.2 APiXC2 DAQ Firmware for ARDUINO DUE (SAM3X8E ARM).

```
int column = 48;
int row = 16;
volatile boolean start_t;
volatile boolean clk;            //sampler debug signal
volatile boolean bValid = 1;

int count = 0;
int i = 0, j = 0;

// TC0 channel 0
void TC0_Handler()
   {
        TC_GetStatus(TC0, 0);
        start_t = !start_t;
        if ( !start_t)
          {
                if (bValid)
                  {
                     Serial.print(count);
                         Serial.println();
                  }
                count = 0;
      }
        }

void TimerStart(Tc *tc, uint32_t channel, IRQn_Type irq, uint32_t divider)
{
        pmc_set_writeprotect(false);
        pmc_enable_periph_clk(irq);
        TC_Configure(tc,channel,TC_CMR_WAVE |
                        TC_CMR_WAVSEL_UP_RC | C_CMR_TCCLKS_TIMER_CLOCK1);
        uint32_t rc = VARIANT_MCK / 2 / divider;
        TC_SetRA(tc, channel, rc >> 1);
        TC_SetRC(tc, channel, rc);
        TC_Start(tc, channel);
        tc->TC_CHANNEL[channel].TC_IER= TC_IER_CPCS | TC_IER_CPAS;
        tc->TC_CHANNEL[channel].TC_IDR=~(TC_IER_CPCS | TC_IER_CPAS);
        NVIC_EnableIRQ(irq);
}

void setup()
{
        //serial port declaration
        Serial.begin(115200);
        //enable pinout
        pinMode (VSR_RST, OUTPUT);
        pinMode (VSR_CLK, OUTPUT);
        pinMode (EN_RST, OUTPUT);
        pinMode (EN_CLK, OUTPUT);
        pinMode (EN_IN, OUTPUT);
        pinMode (TEST_N, OUTPUT);
        //enable idle state
        digitalWrite (VSR_RST,HIGH);
        digitalWrite (VSR_CLK,LOW);
        digitalWrite (EN_RST,HIGH);
        digitalWrite (EN_CLK,LOW);
        digitalWrite (EN_IN,LOW);
        digitalWrite (TEST_N, HIGH);
        //counter port declaration and init state
        pinMode(outOR, INPUT);
        TimerStart(TC0, 0, TC0_IRQn, 10 );
        attachInterrupt(digitalPinToInterrupt(outOR), counter, RISING);
}

void loop()
{
        for (i = 8 ; i < 16 ; i++)
           {
                for (j = 0; j < 48; j++)
                   {
                        while ( Serial.readString() != "?0000:0000\n")
                         {
                               digitalWrite (TEST_N, LOW);
                               delayMicroseconds(20);
                               digitalWrite (TEST_N, HIGH);
                               delay(2000);
                         }
                        Serial.println("Enable␣signal␣catched!");
                        bValid = 0;
                        enable (j,i);
                        Serial.println("Programmed!");
                        bValid = 1;
                  }
           }
```

```
}

void counter()
{
  if ( start_t ) count++;
}
```

A.3 APiXC2 DAQ software.

```
//c++ libraries
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <unistd.h>
#include <ctime>


//ROOT CERC C libraries
#include "TROOT.h"
#include "TApplication.h"
#include "TTree.h"
#include "TBranch.h"
#include "TFile.h"
#include "TH1.h"
#include "TCanvas.h"

//modified boost libraries
#include "TimeoutSerial.h"

using namespace std;
using namespace boost;

unsigned int uColumn = 48;
unsigned int uRow = 16;

//function declaration
inline void f_usage();
inline void f_version();
inline void f_help(unsigned int&, unsigned int&, string& );
inline void f_rest(unsigned int);
inline void f_test(unsigned int&, string& );

int main(int argc, char* argv[])
  {
       //variable declaration
       unsigned int uBaud = 115200; //baude rate
       unsigned int uTime = 30;         //data time
       unsigned int uSkip = 0;
       unsigned int uData;
       unsigned int uLeft;
       unsigned int uWait;
```

```
        int iCh;

        bool bFile = false;
        bool bRoot = false;
        bool bMonitor = false;
        bool bValid;
        bool bNorm = false;
        bool bCycle = false;

        string sRoot;
        string sFname = to_string(time(0));
        string sPort = "ttyUSB0";
        string sData;

        ofstream fout;

        //optarg handler
    while ((iCh = getopt(argc, argv, "n:c:b:p:f:et:r:mvh:g")) != -1)
        switch (iCh)
        {
                 case 'n':
                   bNorm = true;
                   break;
                 case 'c':
                   bCycle = true;
                   break;
         case 'b' :
           uBaud = atoi(optarg);
           break;
         case 'p' :
           sPort = optarg;
           break;
         case 'f' :
           bFile = true;
           sFname = optarg;
           break;
         case 'e' :
           bFile = true;
           break;
         case 't' :
           uTime = atoi(optarg);
           break;
         case 'r' :
           bRoot = true;
           sRoot = optarg;
           break;
         case 'm' :
           bMonitor = true;
           break;
         case 'v' :
           f_version();
```

```
            return 1;
          case 'h' :
            f_help(uBaud, uTime, sPort);
            return 1;
                  case 'g':
                    f_test(uBaud, sPort);
                    return 1;
          default :
            f_usage();
            return 1;
          }

    //output stream
    ostream & dout = (bFile) ? fout : cout;

    //Root initialization
    TApplication app("APiX␣Pulser", &argc, argv);
    gROOT->Reset();
    TTree* tAPIX = new TTree("APiX", "APiX␣pulser");
    TBranch *tData = tAPIX->Branch("counts", &uData);

    //Serial port setting
    TimeoutSerial serial("/dev/"+sPort, uBaud);

    //try opening serial port
    try
    {
        cout << "=========================================================" << endl;
        cout << "||␣␣␣␣APiX␣Pulse␣Reader␣1.1␣FOR␣ARDUINO␣DUE␣␣␣␣␣␣␣␣||" << endl;
        cout << "=========================================================" << endl;
        cout << endl;
        cout << "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;
        cout << "++++++++++++++++++++++␣WARNING!␣+++++++++++++++++++++++" << endl;
        cout << "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;
        cout << "Open␣minicom␣connected␣to␣" << sPort << "␣in␣a␣new
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣terminal," << endl;
        cout << "then␣press␣ENTER␣to␣continue" << endl;
        cin.get();

        cout << "---------------------------------------------------------" << endl;
    cout << "Opening␣serial␣port␣" << sPort;
    cout << ",␣baud␣rate␣" << uBaud << endl;


        if (bNorm && !bCycle)
          {
                cout << "Normal␣mode␣selected" << endl;
        cout << "for␣" << uTime << "␣s\n";
                cout << "for␣one␣pixel" << endl;
        if (bFile)
              {
                sFname += ".dat";
                    cout << "Writing␣output␣to␣" << sFname << endl;
            fout.open(sFname);
                  }
        if (bRoot)
                  {
            sRoot += ".root";
                    cout << "Writing␣output␣to␣" << sRoot << endl;
                  }
          }
        if (!bNorm && bCycle)
          {
                cout << "Cyclical␣mode␣selected" << endl;
        cout << "for␣" << uTime << "␣s\n";
                cout << "for␣one␣pixel" << endl;
        if (bFile)
                  {
                    cout << "Writing␣output␣to␣" << sFname << "␣file␣data
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣set" << endl;
                  }
        if (bRoot)
                  {
                    cout << "Writing␣output␣to␣" << sRoot << "␣ROOT␣file
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data␣set" << endl;
                  }
        }
    if (bNorm && bCycle)
          {
                cout << "Error:␣Unexpected␣choice:␣conflict" << endl;
                cout << "Exiting..." << endl;
                return 1;
          }
    serial.setTimeout(posix_time::seconds(1));  //1 second timeout
    f_rest(2);
    cout << endl;
    }
    catch (boost::system::system_error& e)
    {
        cout << "Error:␣" << e.what() << endl;
        return 1;
    }
    if (bNorm)
     {
        uLeft = uTime;
        uWait = time(0) + uTime;
        //loop for uTime seconds, read and write operation
            serial.writeString("?0000:0000\n");
            f_rest(5);
        while(time(0) <= uWait)
        {
           bValid = false;
    if (uWait-time(0) == uLeft) //Seconds left to exit while
    {
      cout << "..." << uTime-- << " s left...\n";
      uLeft = uTime;
    }
    try //write and read operation
    {
      bValid = true;
          sData = serial.readStringUntil("\n");
      //if not exception was thrown
      if (bValid)
      {
          uData = stoul(sData, nullptr, 10);

      if (bRoot) tAPIX->Fill();       //fill the branch
      else      //else write to stream
          {
             dout << sPort << ":\t";
             dout << uData << endl;
          }
        }
       }

     //catch timeout exception
     catch (const runtime_error& e)
      {
        cout << "\n#" << ++uSkip;
        cout << " Runtime error: " << e.what() << endl;
      }
     //catch timeout exception
     catch (const invalid_argument& e)
       {
         cout << "\n#" << ++uSkip;
         cout << " Invalid argument: " << e.what() << endl;
       }
    }

  serial.close();      //close serial port
  cout << "\nNumber of exception thrown " << uSkip << endl;

  if (bRoot && bValid)     //attach tree to sepcified root file
    {
      char cString[1024];
      strcpy(cString, sRoot.c_str());
      TFile tFout (cString, "RECREATE");
      if (tFout.IsZombie())
         {
           cout << "\nError creating root file\t";
           cout << sRoot << "\nExiting.\n";
           return 1;
         }
```

```
                tFout.cd();
                tAPIX->Write();
                cout << "\nRoot␣file␣successfully␣saved.\n";
                tFout.Close();
              }

          return 0;
     }

     if (bCycle)
       {
          for (int i = 0 ; i < uColumn ; i++)
            {
               for (int j = 0 ; j < uRow ; j++)
                 {
                    if (bFile)
                         {
                              sFname += to_string(i);
                              sFname += ".";
                              sFname += to_string(j);
                              sFname += ".dat";
                              fout.open(sFname);
                         }
                     if (bRoot)
                         {
                              sRoot += to_string(i);
                              sRoot += ".";
                              sRoot += to_string(j);
                              sRoot += ".root";
                         }
            cout << "-->␣Starting␣for␣Test␣dot␣(" << i << "," << j << ")"
                                                                    << endl;
                    uLeft = uTime;
            uWait = time(0) + uTime;
            serial.writeString("?0000:0000\n");
                    cout << "Wait␣for␣5␣seconds" << endl;
                 f_rest(5);
            //loop for uTime seconds, read and write operation
            while(time(0) <= uWait)
               {
                     bValid = false;
                  if (uWait-time(0) == uLeft)
                      {
                          cout << "..." << uTime-- << "␣s␣left...\n";
                          uLeft = uTime;
                      }
                  try //write and read operation
                     {
                         bValid = true;
                              sData = serial.readStringUntil("\n");
                         //if not exception was thrown
                    if (bValid)
                    {
                         uData = stoul(sData, nullptr, 10);
                      if (bRoot) tAPIX->Fill();
                      else    //else write to stream
                        {
                         dout << sPort << ":\t";
                         dout << uData << endl;
                        }
                     }
                }

             //catch timeout exception
             catch (const runtime_error& e)
              {
                     cout << "\n#" << ++uSkip;
                 cout << "␣Runtime␣error:␣" << e.what() << endl;
               }
             //catch timeout exception
             catch (const invalid_argument& e)
              {
                 cout << "\n#" << ++uSkip;
                 cout << "␣Invalid␣argument:␣" << e.what() << endl;
               }
             }

             if (bRoot && bValid) //attach tree to specified root file
           {
             char cString[1024];
             strcpy(cString, sRoot.c_str());
             TFile tFout (cString, "RECREATE");
             if (tFout.IsZombie())
               {
                  cout << "\nError␣creating␣root␣file\t";
                  cout << sRoot << "\nExiting.\n";
                  return 1;
               }
             tFout.cd();
             tAPIX->Write();
             cout << "\nRoot␣file␣successfully␣saved.\n";
             tFout.Close();
           }
                 cout << "-->␣Go␣to␣the␣next␣pixel..." << endl;
                 f_rest(1);
             }
         }
       serial.close();     //close serial port
   cout << "\nNumber␣of␣exception␣thrown␣" << uSkip << endl;
   }
   return 0;
}
```

```
//executable usage
inline void f_help(unsigned int& uBaud, unsigned int& uTime, string& sPort)
{
   cout << "Usage:␣serial␣[OPTION]␣[FILE_NAME]\n\n";
   cout << "Option:\n";
   cout << "\t-h\tPrint␣this␣message␣and␣exit\n";
   cout << "\t-v\tPrint␣version\n\n";
   cout << endl;
   cout << "\t-n\tNormal␣execution␣mode␣(1␣pixel␣measurement)" << endl;
   cout << "\t-c\tCyclical␣execution␣mode␣(all␣pixels␣measurement)" << endl;
   cout << "\t-b\tSet␣baud␣rate,␣default␣" << uBaud << endl;
   cout << "\t-p\tSet␣tty␣serial␣port,␣default␣" << sPort << endl;
   cout << "\t-e\tSet␣file␣output,␣creates␣<epoch>.dat\n";
   cout << "\t-t\tSet␣serial␣port␣life␣span,␣default␣" << uTime << "s\n";
   cout << "\t-f\tSave␣output␣to␣.dat␣file,␣pass␣<filename>\n";
   cout << "\t-r\tSave␣to␣.root␣file,␣pass␣<filename>\n";
   cout << "\t-m\tEnable␣live␣monitoring␣(not␣implemented␣yet)\n";
   cout << endl;
   cout << "\t-g\tPin␣test␣utility" << endl;
}

inline void f_usage()
{
    cout << "Usage:␣serial␣[OPTION]␣[FILE_NAME]\n\n";
    cout << "Type␣\"serial␣-h\"␣for␣more␣option\n";
}

inline void f_version()
{
    cout << "\t␣serialapix␣for␣arduino\t␣v1.1\n";
    cout << "==========================================================" << endl;
    cout << "||␣␣␣␣APiX␣Pulse␣Reader␣1.1␣FOR␣ARDUINO␣DUE␣␣␣␣␣␣␣||" << endl;
    cout << "==========================================================" << endl;
    cout << endl;
    cout << "Author:␣Giuseppe␣Spadoni\n";
    cout << "Libraries:␣Federico␣Terraneo␣and␣Tommaso␣Boschi,␣\n";
    cout << "␣distribuited␣under␣the␣Boost␣Software␣Licence,␣v␣1.58.0,\n";
    cout << "All␣Rights␣Reserved.␣\n";
}

//rest for s sec
inline void f_rest(unsigned int s)
{
    clock_t clkt = clock()+CLOCKS_PER_SEC*s;
    while (clkt >= clock()) {}
}

inline void f_test(unsigned int& uBaud, string& sPort)
        {
    TimeoutSerial serial("/dev/"+sPort, uBaud);
    //try opening serial port
    try
    {
        cout << "====================================================" << endl;
        cout << "||␣␣␣␣␣␣␣␣␣␣␣APiX␣PIN␣TEST␣FOR␣ARDUINO␣DUE␣␣␣␣␣␣␣␣␣||" << endl;
        cout << "====================================================" << endl;
        cout << endl;
        cout << "+++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;
        cout << "+++++++++++++++++++++++␣WARNING!␣+++++++++++++++++++++" << endl;
        cout << "+++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;
        cout << "Open␣minicom␣connected␣to␣" << sPort << "␣in
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣a␣new␣terminal," << endl;
        cout << "to␣check␣input␣signal,␣then␣press␣ENTER␣to␣continue" << endl;
        cin.get();

        cout << "-----------------------------------------------------" << endl;
    cout << "Opening␣serial␣port␣" << sPort;
    cout << ",␣baud␣rate␣" << uBaud << endl;
    serial.setTimeout(posix_time::seconds(1));   //1 second timeout
    f_rest(2);
    cout << endl;
    }

   catch (boost::system::system_error& e)
    {
        cout << "Error:␣" << e.what() << endl;
    }
   for (int i = 0 ; i < uColumn ; i++)
           {
         for (int j = 0 ; j < uRow ; j++)
                 {
               cout << "--->␣Starting␣for␣Test␣dot␣(" << i << "," << j << ")"
                                                                 << endl;
               serial.writeString("?0000:0000\n");
                       cout << "Wait␣for␣5␣seconds" << endl;
                   f_rest(5);
               //loop for pin test
                       cout << "press␣enter␣for␣the␣next␣spot." << endl;
                   cin.get();
                       cout << "--->␣Go␣to␣the␣next␣pixel..." << endl;
                       f_rest(1);
                    }
           }
        serial.close();       //close serial port
}
```

A.4 APiXC1 Multipurpose firmware for ARDUINO DUE (SAM3X8E ARM)

This firmware is written with the scope to program the enabling pattern of both chips, use the internal FIFOs of APiXC1 and also configure the crosstalk coincidence.

```
// Serial baud rate: default value
//int baudrate = 9600;
int baudrate = 19200;

// Digital control signal configuration
int TESTN = 49;

//Vertical shift registers: shift the active row each clock
int VSR_CK = 51;
int VSR2_CK = 45;
int VSR_RN = 53;

int FIFO_RN = 41;
int FIFO_SEL = 39;
int FIFO_TX = 37;
int MEM_RN = 35;

int EN_CK = 27;
int EN_RN = 25;
int EN_IN = 23;

//outputs
int FIFO_1 = 50;
int FIFO_2 = 48;
int FIFO_3 = 46;
int FIFO_4 = 44;
int FIFO_5 = 42;
int FIFO_6 = 40;
int FIFO_7 = 38;
int FIFO_8 = 36;


// SPAD enable mode: 1: ROI, 2: 2 SPADs, 3: 2 rows
int mode;

// Active pixel ROI (Region of Interest)
int col_min;
int col_max;
int row_min;
int row_max;

// Integration time in microseconds
int tint;

// Run mode: 1: continuous, 2 integr. + readout
int runmode;

// Frame
int frame;

#define ROWS 8
```

```
#define COLUMNS 96

union Data {
    unsigned int  i[ROWS*COLUMNS];      // analog input array value
    char          s[4*ROWS*COLUMNS];    // 4 bytes for each integer
} m;


String inString = "";     // string to hold input

// Reads input string, ended by \n character
int read_inString() {
  inString = "";
  bool eos = false;
  while ( true ) {
    // Read serial input:
    while (Serial.available()) {
      int inChar = Serial.read();
      if (isDigit(inChar)) {
        inString += (char)inChar;
      }
      if (inChar == '\n') {
        eos = true;
        break;
      }
    }
    if (eos) {
      return 1;
      break;
    }
  }
}

// Reads configuration parameters from serial port
void read_cfgparams() {

    //mode
    int code = read_inString();
    if (code == 1) {
        mode = inString.toInt();
    } else {
        mode = 1;
    }

    //col_min
    code = read_inString();
    if (code == 1) {
        col_min = inString.toInt();
    } else {
        col_min = 24;
    }
    //col_max
    code = read_inString();
    if (code == 1) {
        col_max = inString.toInt();
    } else {
        col_max = 24;
    }

    //row_min
    code = read_inString();
    if (code == 1) {
        row_min = inString.toInt();
    } else {
        row_min = 8;
    }

    //row_max
    code = read_inString();
    if (code == 1) {
        row_max = inString.toInt();
    } else {
        row_max = 8;
    }

    //tint
    code = read_inString();
    if (code == 1) {
        tint = inString.toInt();
    } else {
        tint = 8;
    }

    //runmode
    code = read_inString();
    if (code == 1) {
        runmode = inString.toInt();
    } else {
        runmode = 1;
    }

    //frame

    code = read_inString();
    if (code == 1) {
        frame = inString.toInt();
    } else {
        frame = 100;
    }

}
```

```
// Sends configuration message via serial port
void send_cfgmsg() {
    Serial.print("Setting OK. Mode: ");
    Serial.print(mode);
    Serial.print(" Columns: ");
    Serial.print(col_min);
    Serial.print(" - ");
    Serial.print(col_max);
    Serial.print(" Rows: ");
    Serial.print(row_min);
    Serial.print(" - ");
    Serial.print(row_max);
    Serial.print(" Tint: ");
    Serial.print(tint);
    Serial.print(" Runmode: ");
    Serial.print(runmode);
    Serial.print(" Frame: ");
    Serial.println(frame);

}

// Configure enable register - ROI (Region Of Interest) selection
void EN_config_ROI() {
   // Disable all SPADs
  digitalWrite(TESTN, LOW);
  // VSR reset
  digitalWrite(VSR_RN, LOW);
  digitalWrite(VSR_RN, HIGH);
  // Loop on rows and columns
  for (int i=0; i < 16; i++){     // for row
      // Goes to next line
      digitalWrite(VSR_CK, HIGH);
      digitalWrite(VSR_CK, LOW);
      // Resets the enable register of the current line
      digitalWrite(EN_RN, LOW);
      digitalWrite(EN_RN, HIGH);
      // Loads the enable reg. of the current line
      for (int j=0; j < 48; j++){ // for col
          if ((i >= row_min) && (i <= row_max)
            && (j >= col_min) && (j <= col_max)) {
              digitalWrite(EN_IN, HIGH);
              digitalWrite(EN_CK, HIGH);
              digitalWrite(EN_IN, LOW);
              digitalWrite(EN_CK, LOW);
          } else {
              digitalWrite(EN_CK, HIGH);
              digitalWrite(EN_CK, LOW);
          }
      }  // for col
  }  // for row
  // Enable all SPADs
  digitalWrite(TESTN, HIGH);
}

// Configure enable register - 2 SPAD selection
void EN_config_2S() {
  // Disable all SPADs
  digitalWrite(TESTN, LOW);
  // VSR reset
  digitalWrite(VSR_RN, LOW);
  digitalWrite(VSR_RN, HIGH);
  // Loop on rows and columns
  for (int i=0; i < 16; i++){     // for row
      // Goes to next line
      digitalWrite(VSR_CK, HIGH);
      digitalWrite(VSR_CK, LOW);
      // Resets the enable register of the current line
      digitalWrite(EN_RN, LOW);
      digitalWrite(EN_RN, HIGH);
      // Loads the enable reg. of the current line
      for (int j=0; j < 48; j++){ // for col
          if (((i == row_min) && (j == col_min))
            || ((i == row_max) && (j == col_max))) {
              digitalWrite(EN_IN, HIGH);
              digitalWrite(EN_CK, HIGH);
              digitalWrite(EN_IN, LOW);
              digitalWrite(EN_CK, LOW);
          } else {
              digitalWrite(EN_CK, HIGH);
              digitalWrite(EN_CK, LOW);
          }
      }  // for col
  }  // for row
  // Enable all SPADs
  digitalWrite(TESTN, HIGH);
}

// Configure enable register - 2 rows selection
void EN_config_2R() {
  // Disable all SPADs
  digitalWrite(TESTN, LOW);
  // VSR reset
  digitalWrite(VSR_RN, LOW);
  digitalWrite(VSR_RN, HIGH);
  // Loop on rows and columns
  for (int i=0; i < 16; i++){     // for row
      // Goes to next line
      digitalWrite(VSR_CK, HIGH);
      digitalWrite(VSR_CK, LOW);
      // Resets the enable register of the current line
      digitalWrite(EN_RN, LOW);
      digitalWrite(EN_RN, HIGH);
      // Loads the enable reg. of the current line
      for (int j=0; j < 48; j++){ // for col
          if (((i == row_min) && (j >= col_min) && (j <= col_max)) ||
              ((i == row_max) && (j >= col_min) && (j <= col_max))) {
              digitalWrite(EN_IN, HIGH);
              digitalWrite(EN_CK, HIGH);
              digitalWrite(EN_IN, LOW);
              digitalWrite(EN_CK, LOW);
          } else {
              digitalWrite(EN_CK, HIGH);
              digitalWrite(EN_CK, LOW);
          }
      }  // for col
  }  // for row
  // Enable all SPADs
  digitalWrite(TESTN, HIGH);
}

// Selects active row and row2 for coincidence
void VSR_goto(int row, int row2) {
  // VSR reset
  digitalWrite(VSR_RN, LOW);
  digitalWrite(VSR_RN, HIGH);
  // sends n=row VSR clock pulses
  for (int i=0; i < (row+1); i++){
      digitalWrite(VSR_CK, HIGH);
      digitalWrite(VSR_CK, LOW);
  }
  // sends n=row2 VSR2 clock pulses
  for (int i=0; i < (row2+1); i++){
      digitalWrite(VSR2_CK, HIGH);
      digitalWrite(VSR2_CK, LOW);
  }
}

// Initialize readout: reset FIFO and memory
void init_RO() {
  digitalWrite(FIFO_RN, LOW);
  digitalWrite(FIFO_RN, HIGH);
  digitalWrite(MEM_RN, LOW);
  digitalWrite(MEM_RN, HIGH);
}

// Integration: optionally include a test pulse
void integrate() {
  digitalWrite(TESTN, HIGH); // Activates SPADs
  delayMicroseconds(tint); // Integration time
}

// Transfer data from memories to output register
```

```
void data_TX() {
  digitalWrite(FIFO_SEL, LOW); // Enables transfer from memory
  digitalWrite(FIFO_TX, HIGH); // Transfers data
  digitalWrite(FIFO_TX, LOW);
   // Back to output register transfer configuration
  digitalWrite(FIFO_SEL, HIGH);
  digitalWrite(TESTN, LOW);
}

// Data output
void data_OUT(int * local_fifo_matrix) {
    digitalWrite(MEM_RN, LOW); // Resets memory
    digitalWrite(MEM_RN, HIGH); //
    for (int i=0; i < 96; i++){ // Data output clock (48 x 2)
        digitalWrite(FIFO_TX, HIGH);
        digitalWrite(FIFO_TX, LOW);

        if ( i<48) {
          local_fifo_matrix[0*96 + i + 48] = digitalRead(FIFO_1);
          local_fifo_matrix[1*96 + i + 48] = digitalRead(FIFO_2);
          local_fifo_matrix[2*96 + i + 48] = digitalRead(FIFO_3);
          local_fifo_matrix[3*96 + i + 48] = digitalRead(FIFO_4);
          local_fifo_matrix[4*96 + i + 48] = digitalRead(FIFO_5);
          local_fifo_matrix[5*96 + i + 48] = digitalRead(FIFO_6);
          local_fifo_matrix[6*96 + i + 48] = digitalRead(FIFO_7);
          local_fifo_matrix[7*96 + i + 48] = digitalRead(FIFO_8);
         } else {

        local_fifo_matrix[0*96 + i - 48] = digitalRead(FIFO_1);
        local_fifo_matrix[1*96 + i - 48] = digitalRead(FIFO_2);
        local_fifo_matrix[2*96 + i - 48] = digitalRead(FIFO_3);
        local_fifo_matrix[3*96 + i - 48] = digitalRead(FIFO_4);
        local_fifo_matrix[4*96 + i - 48] = digitalRead(FIFO_5);
        local_fifo_matrix[5*96 + i - 48] = digitalRead(FIFO_6);
        local_fifo_matrix[6*96 + i - 48] = digitalRead(FIFO_7);
        local_fifo_matrix[7*96 + i - 48] = digitalRead(FIFO_8);
        }

      //digital acquisition 8x96 clock lines
    }
}

// Generates integration + Read Out sequence
void gen_int_RO_sequence() {
    //initialize the matrix
    int row = 0;
    int col = 0;

    int local_fifo_matrix[96*8];

    init_RO();
    //while (true) {
      /*
        if (Serial.available() > 0) {
            int code = read_inString();
            if (code == 1) {
                if (inString == "0") {
                    Serial.println("Generation terminated");
                    break;
                }
            }
        }
        */

        int n = 0;
        while (n < frame) {
            integrate();
            data_TX();
            data_OUT(local_fifo_matrix);

            for (row = 0; row < 8; ++row) {
                for (col = 0; col < 96; ++col) {
                    m.i[96*row + col] += *(local_fifo_matrix + 96*row + col);
                }
            }

            ++n;
        }

        //return return_matrix through serial
        Serial.write(m.s,4*ROWS*COLUMNS);
        Serial.println("Generation␣terminated");

        for (row = 0; row < 8; ++row) {
           for (col = 0; col < 96; ++col) {
              m.i[96*row + col] = 0;
           }
        }

    //}
}

// Free running SPADs (for external readout)
void free_running() {
  digitalWrite(TESTN, HIGH); // enable SPADs
  while(true) {
    if (Serial.available() > 0) {
      int code = read_inString();
      if (code == 1) {
        if (inString == "0") {
          Serial.println("Free␣running␣execution␣terminated");
          digitalWrite(TESTN, LOW); // disable SPADs
          break;
        }
      }
    }
    delay(1); // Do nothing, just wait
  } // while
}

void setup() {
    pinMode(VSR_RN, OUTPUT);
    pinMode(VSR_CK, OUTPUT);
    pinMode(VSR2_CK, OUTPUT);
    pinMode(TESTN, OUTPUT);
    pinMode(FIFO_RN, OUTPUT);
    pinMode(FIFO_SEL, OUTPUT);
    pinMode(FIFO_TX, OUTPUT);
    pinMode(MEM_RN, OUTPUT);
    pinMode(EN_CK, OUTPUT);
    pinMode(EN_RN, OUTPUT);
    pinMode(EN_IN, OUTPUT);

    pinMode(FIFO_1, INPUT);
    pinMode(FIFO_2, INPUT);
    pinMode(FIFO_3, INPUT);
    pinMode(FIFO_4, INPUT);
    pinMode(FIFO_5, INPUT);
    pinMode(FIFO_6, INPUT);
    pinMode(FIFO_7, INPUT);
    pinMode(FIFO_8, INPUT);

    // Signal initialization
    digitalWrite(VSR_RN, HIGH);
    digitalWrite(VSR_CK, LOW);
    digitalWrite(VSR2_CK, LOW);
    digitalWrite(TESTN, LOW);
    digitalWrite(FIFO_RN, HIGH);
    digitalWrite(FIFO_SEL, HIGH);
    digitalWrite(FIFO_TX, LOW);
    digitalWrite(MEM_RN, HIGH);
    digitalWrite(EN_CK, LOW);
    digitalWrite(EN_RN, HIGH);
    digitalWrite(EN_IN, LOW);

    // Enable register configuration
    //EN_config_ROI();
    //EN_config_2S();
    // VSR is positioned on the row row_min
    //VSR_goto(row_min, row_max);
    //VSR_goto(S1_row, S2_row);
    // start serial port
    Serial.begin(baudrate);
    //Serial.println("Seriale configurata");
    delay(100);
}

void loop() {

  delay(10);
  // if we get a valid byte, read analog ins:
  if (Serial.available() > 0) {
    read_cfgparams(); // Reads config. parameters from serial port
    send_cfgmsg();      // Sends feedback message with config. parameters

    // Enable register configuration
    if (mode == 1) {
        EN_config_ROI();
    } else if (mode == 2) {
        EN_config_2S();
    } else if (mode == 3) {
        EN_config_2R();
    }

    // VSR is positioned on the row row_min and VSR2 on the row row_max
    VSR_goto(row_min, row_max);

    // Starts readout sequence
    //(can be stopped by sending "0\n" on the serial port)
    delay(1);
    if (runmode == 2) { // integration + read out sequence
      gen_int_RO_sequence();
    }
    if (runmode == 1) {  // Free running sequence
      free_running();
    }

    // stops and restarts serial to clear buffer
    delay(100);
    Serial.end();
    Serial.begin(baudrate);
  }

}
```

A.5 APiX Matrix Generator

This software, written in OCTAVE, is able to manage the ARDUINO DUE and two power supplies, connected with GPIB-B to USB controller made by National Instruments. The initialization of the external devices are written in 5 sources files, not included in this section.

A.5.1 Matrix generation with *All-On* algorithm

```
% Test chip APIX Sandwich - Entire Submatrix On
```

```
%
% Set continuous running and measure count rate
% transerring from Arduino the data contained into the internal
% memory of each pixel

clear all

% Serial port initialization (arduino)
A = ARDUINO_init('COM3');
% GPIB-USB port initialization (Agilent)
P_supply = POWERsupply_init('ni', 0, 14);
% GPIB-USB port initialization (Agilent)
P_bias = POWERbias_init('ni', 0, 13);

% Configuration parameters values
p = struct('col_min',5,'col_max',5,'row_min',5,'row_max',5,...
    'mode',1,'tint',1000,'runmode',2,'frame',100);
% mode (1: ROI, 2: 2 SPADS, 3: 2 ROWS )
% tint Integration time in microseconds
% runmode: 1 = continuous; 2 = integr + readout


Vn=-2; %Bypass the upper limit of the supplies
% Vmediano=18.29+Vn; %S1 breakdown(uncomment to use)
Vmediano=21.90+Vn;   %S2 breakdown(uncomment to use)

VAPPL(P_bias,0,Vn);

clear c;
clear M;
indx=1;

%row and column indexes (entire S1 submatrix selected)
a=0;
b=7;
c=0;   %3  42
d=47;  %5  44

% Nframe=50;
Nframe=50000;

for k = 1:21     %loop for the download

    clear counts
    clear Max
    k
    p.row_min = a+8;%+8;
    p.row_max = b+8;%+8;
    p.col_min = c;
    p.col_max = d;
    if k == 1
```

```
            p.frame=5;
        else
            p.frame=Nframe;
        end
        %Send parameters to ARDUINO
        msg=CONFIG(A,p);
        %Print the sent parameters
        fprintf(msg);
        %Supplies parameters
        Vex=Vmediano+2;  %1V excess bias
        VAPPL(P_bias,Vex,Vn);
        pause(0.1);
        for y=1:768
            Mtx=fread(A,4,'char');
            counts(y)=Mtx(1)+16^2*Mtx(2)+16^4*Mtx(3)+16^6*Mtx(4);
%S1
%           counts=Mtx(1)+16^2*Mtx(2)+16^4*Mtx(3)+16^6*Mtx(4)
%S2
%           counts(y,x)=Mtx(1,x)+16^2*Mtx(2,x)+16^4*Mtx(3,x)+16^6*Mtx(4,x);  %S2
        end
        M(:,k)=counts;
        pause(0.1);
        msg = fscanf(A);
        fprintf(msg);

    end     % for k

%Count rate generation
M_f=M(:,2:k);
counts_sum=sum(M_f,k-1);

%Map generator
for y=1:16
    for x=1:48
        Mappa(y,x)=M(x+48*(y-1),20);
        Mappa(y,x)=counts_sum(x+48*(y-1));
    end
end

% Mappa_ok=Mappa(1:8,1:48);        %S1
Mappa_ok=Mappa(9:16,1:48);         %S2

%Saving into a text file
save -ascii 'file_name.txt' Mappa_ok;

% Switch off generators and close GPIB communication
POWER_off(P_supply);
POWER_off(P_bias);

% Cleans buffer and closes serial communication with Arduino
ARDUINO_off(A);
```

A.5.2 Matrix generation with *Single-On* algorithm

```
clear all
instrfind
% Serial port initialization (arduino)
A = ARDUINO_init('COM6');
% GPIB-USB port initialization (Agilent) alimentatore per la curcuiteria
P_supply = POWERsupply_init('ni', 0, 14);
% GPIB-USB port initialization (Agilent) alimentatore per Vbias SPAD
P_bias = POWERbias_init('ni', 0, 20);

% Configuration parameters values
p = struct('col_min',5,'col_max',5,'row_min',5,'row_max',5,...
    'mode',1,'tint',10,'runmode',2,'frame',100);
% mode (1: ROI, 2: 2 SPADS, 3: 2 ROWS )
% tint Integration time in microseconds
% runmode: 1 = continuous; 2 = integr + readout

% initialization Debug
% msg=CONFIG(A,p);
% fprintf(msg);

Vn=-2;  %Bypass supply limit

Vmediano=18.29+Vn;    %Sandwich
% Vmediano=21.89+Vn; %S2
% Vmediano=17.88+Vn; %S1

VAPPL(P_bias,0,Vn);

clear c;
clear M;
indx=1;

%init matrix
M_f=zeros(768,2);
counts_sum=zeros(768);

Nframe=50000;
% Nframe=5;

for i=0:7          %FIFO shifting cycle
    for j=42:44    %23     42:44    3:5
        for k = 1:3     %Download from Arduino
        clear counts
        clear Max
        k
        p.row_min = i;%+8;
        p.row_max = i;%+8;
        p.col_min = j;
        p.col_max = j;
        if k == 1
```

```
                p.frame=5;
            else
                p.frame=Nframe;
            end
            %send parameters to Arduino
            msg=CONFIG(A,p);
            %debug parameter
            fprintf(msg);
            Vex=Vmediano+3;   %3V excess bias

            VAPPL(P_bias,Vex,Vn);
            pause(0.1);
            for y=1:768
                Mtx=fread(A,4,'char');
                counts(y)=Mtx(1)+16^2*Mtx(2)+16^4*Mtx(3)+16^6*Mtx(4);
%S1
%               counts=Mtx(1)+16^2*Mtx(2)+16^4*Mtx(3)+16^6*Mtx(4)
%S2
%               counts(y,x)=Mtx(1,x)+16^2*Mtx(2,x)+16^4*Mtx(3,x)+16^6*Mtx(4,x);%S2
            end
           M(:,k)=counts;
            pause(0.1);
            msg = fscanf(A);
            fprintf(msg);
            end      % for k
        %count rate generation
        %S1
        M_f=M(i*48+j+1,2:3);
        counts_sum(i*48+j+1)=sum(M_f,2);
        %S2
%       M_f=M((i+8)*48+j+1,2:3);
%       counts_sum((i+8)*48+j+1)=sum(M_f,2);
        end
end

%Map generation
for y=1:16
    for x=1:48
        Mappa(y,x)=counts_sum(x+48*(y-1));
    end
end

Mappa_ok=Mappa(1:8,1:48);
% Mappa_ok=Mappa(9:16,1:48);

%save into a text file
save -ascii 'file_name.txt' Mappa_ok;

% Switch off generators and close GPIB communication
%POWER_off(P_supply);
POWER_off(P_bias);
```

```
% Cleans buffer and closes serial communication with Arduino
ARDUINO_off(A);
```

A.6 APiXC1 DC-Sweep software

This software program the supplies to generate automatically the V-Counts Sweep for a given matrix pattern. Count rates are acquired by a NI-DAQ device in free-running mode.

```
% Serial port initialization (arduino)
A = ARDUINO_init('COM5');
% GPIB–USB port initialization (Agilent)
P_supply = POWERsupply_init('ni', 0, 14);
% GPIB–USB port initialization (Agilent)
P_bias = POWERbias_init('ni', 0, 13);
%Counter initialization
if ~exist('sdaq')
    sdaq = NI_Count_init();
end
% Configuration parameters values
p = struct('col_min',13,'col_max',13,'row_min',8,'row_max',8,...
    'mode',2,'tint',10,'runmode',1);
% mode (1: ROI, 2: 2 SPADS, 3: 2 ROWS )
% tint Integration time in microseconds
% runmode: 1 = continuous; 2 = integr + readout
% msg=CONFIG(A,p);
% fprintf(msg);

% Main loop – voltage sweep and count acquisition
%
% % %Vsteps submatrix 1
Vstart = 16.27;        %15.00
Vstop = 16.47;         %17.00
deltaV = 0.01;         %0.01

Vn=−2;   %Bypass supplies limit
VAPPL(P_bias,0,Vn);

% %Vsteps submatrix 2
% Vstart = 20.10;        %Vstart = 20.00;
% Vstop = 20.90;         %Vstop = 21.00;
% deltaV = 0.1;          %deltaV = 0.01;

Vsteps = (Vstop − Vstart)/deltaV + 1;
clear c;
V = [];
figure(1);

clear M;
indx=1;


for i = 0:0%7 % row
```

```
    for j = 19:47%19:47 % column
        p.row_min = i;%+8;
        p.row_max = i;%+8;
        p.col_min = j;
        p.col_max = j;

        msg=CONFIG(A,p);
        fprintf(msg);

        for k = 1:Vsteps
            V(k) = Vstart + (k-1)*deltaV;
            VAPPL(P_bias,V(k),Vn);
            pause(0.1);
            [counts, dt] = NI_Count_edges(sdaq, 1000);
            fprintf('Voltage: %5.2f, Counts: %d, time %d\n', V(k)-Vn, counts, dt);
            c(k) = counts/dt*1000;
            %plot(V,c);
            %data generation
            M(1,1)=0;
            M(1,2)=0;
            M(1,k+2)=V(k)-Vn;
            M(indx+1,1)=i+1;
            M(indx+1,2)=j+1;
            M(indx+1,k+2) = counts;
        end % for k

        semilogy(V,c);
        hold on
        indx=indx+1;
        fprintf(A, '0'); % sends termination key
        % Reads confirmation message from system
        pause(0.1);
        msg = fscanf(A);
        fprintf(msg);
   end    % for j

end  % for i

save -ascii 'V_Sweep.txt' M;

% Switch off generators and close GPIB communication
POWER_off(P_supply);
POWER_off(P_bias);

% Cleans buffer and closes serial communication with Arduino
ARDUINO_off(A);
```

A.7 APiXC1 Crosstalk DAQ software (NI-DAQ external counter)

```
% Serial port initialization (arduino)
```

```
A = ARDUINO_init('COM5');
% GPIB-USB port initialization (Agilent)
P_supply = POWERsupply_init('ni', 0, 14);
% GPIB-USB port initialization (Agilent)
P_bias = POWERbias_init('ni', 0, 13);
%Counter initialization
if ~exist('sdaq')
    sdaq = NI_Count_init();
end

% Configuration parameters values
p = struct('col_min',13,'col_max',13,'row_min',8,'row_max',8,...
    'mode',2,'tint',10,'runmode',1);
% mode (1: ROI, 2: 2 SPADS, 3: 2 ROWS )
% tint Integration time in microseconds
% runmode: 1 = continuous; 2 = integr + readout
%msg=CONFIG(A,p);
%fprintf(msg);

Vn=-2; %supply limit bypass
Vmediano=21.84+Vn;
VAPPL(P_bias,0,Vn);

clear c;
clear M;
indx=1;

%emitter pixel
row=11;
col=1;

for k=1:3 %measure repetition
    k
    for i=8:10    %S2
        for j = 0:23  %23 Seleziono solamente i pixel grandi
            %Emitter DCR
            p.row_max=row;
            p.col_max=col;
            p.row_min=row;
            p.col_min=col;

            msg=CONFIG(A,p);
            fprintf(msg);

            Vex=Vmediano+3;  %3V excess bias

            VAPPL(P_bias,Vex,Vn);
            pause(0.1);
            [counts, dt] = NI_Count_edges(sdaq, 1000);
            fprintf('Voltage:␣%5.2f,␣Counts:␣%d,␣time␣%d\n', Vex-Vn, counts, dt
            c=counts/dt*1000;
```

```
M(1,1)=0;
M(1,2)=0;
%data generation
M(1,3)=Vex-Vn;
M(indx+1,1)=i+1;
M(indx+1,2)=j+1;
M(indx+1,3) = counts;

 fprintf(A, '0'); % sends termination key
 % Reads confirmation message from system
 pause(0.1);
 msg = fscanf(A);
 fprintf(msg);



 %Revelator DCR
 p.row_max=i;
 p.col_max=j;
 p.row_min=i;
 p.col_min=j;

 msg=CONFIG(A,p);
 fprintf(msg);

 Vex=Vmediano+3;  %3V excess bias

 VAPPL(P_bias,Vex,Vn);
 pause(0.1);
 [counts, dt] = NI_Count_edges(sdaq, 1000);
 fprintf('Voltage:␣%5.2f,␣Counts:␣%d,␣time␣%d\n', Vex-Vn, counts, dt);
 c=counts/dt*1000;

M(indx+1,4) = counts;

 fprintf(A, '0'); % sends termination key
 % Reads confirmation message from system
 pause(0.1);
 msg = fscanf(A);
 fprintf(msg);

 %Coincidence rate
 p.row_max=row;
 p.col_max=col;
 p.row_min=i;
 p.col_min=j;

 msg=CONFIG(A,p);
 fprintf(msg);

 Vex=Vmediano+3;  %3V excess bias
```

```
            VAPPL(P_bias,Vex,Vn);
            pause(0.1);
            [counts, dt] = NI_Count_edges(sdaq, 100000);
            fprintf('Voltage:␣%5.2f,␣Counts:␣%d,␣time␣%d\n', Vex-Vn, counts, dt
            c=counts/dt*1000;

            M(indx+1,5) = counts;

            indx=indx+1;

            fprintf(A, '0'); % sends termination key
            % Reads confirmation message from system
            pause(0.1);
            msg = fscanf(A);
            fprintf(msg);

        end  % for j

    end  % for i


    for i=12:15
            for j = 0:23
            % DCR emitter
            p.row_max=row;
            p.col_max=col;
            p.row_min=row;
            p.col_min=col;

            msg=CONFIG(A,p);
            fprintf(msg);

            Vex=Vmediano+3;  %3V excess bias

            VAPPL(P_bias,Vex,Vn);
            pause(0.1);
            [counts, dt] = NI_Count_edges(sdaq, 1000);
            fprintf('Voltage:␣%5.2f,␣Counts:␣%d,␣time␣%d\n', Vex-Vn, counts, dt
            c=counts/dt*1000;

            M(1,1)=0;
            M(1,2)=0;
            M(1,3)=Vex-Vn;
            M(indx+1,1)=i+1;
            M(indx+1,2)=j+1;
            M(indx+1,3) = counts;

            fprintf(A, '0'); % sends termination key
            % Reads confirmation message from system
            pause(0.1);
            msg = fscanf(A);
        fprintf(msg);
        %DCR revelator
        p.row_max=i;
        p.col_max=j;
        p.row_min=i;
        p.col_min=j;

        msg=CONFIG(A,p);
        fprintf(msg);

        Vex=Vmediano+3;  %3V excess bias

        VAPPL(P_bias,Vex,Vn);
        pause(0.1);
        [counts, dt] = NI_Count_edges(sdaq, 1000);
        fprintf('Voltage:␣%5.2f,␣Counts:␣%d,␣time␣%d\n', Vex-Vn, counts, dt);
        c=counts/dt*1000;
        M(indx+1,4) = counts;

        fprintf(A, '0'); % sends termination key
        % Reads confirmation message from system
        pause(0.1);
        msg = fscanf(A);
        fprintf(msg);

        %Coincidence measure
        p.row_max=row;
        p.col_max=col;
        p.row_min=i;
        p.col_min=j;
        msg=CONFIG(A,p);
        fprintf(msg);

        Vex=Vmediano+3;  %3V excess bias

        VAPPL(P_bias,Vex,Vn);
        pause(0.1);
        [counts, dt] = NI_Count_edges(sdaq, 100000);
        fprintf('Voltage:␣%5.2f,␣Counts:␣%d,␣time␣%d\n', Vex-Vn, counts, dt);
        c=counts/dt*1000;
        M(indx+1,5) = counts;

        indx=indx+1;
        fprintf(A, '0'); % sends termination key
        % Reads confirmation message from system
        pause(0.1);
        msg = fscanf(A);
        fprintf(msg);

        end  % for j
end  % for i
end % end k

save -ascii 'T25_Chip1_Sub2_K_Row4_Col2.txt' M;

% Switch off generators and close GPIB communication
POWER_off(P_supply);
POWER_off(P_bias);

% Cleans buffer and closes serial communication with Arduino
ARDUINO_off(A);
```

Appendix B

Simulations with PENELOPE

To simulate the rate of electron-events and the geometrical acceptance of our detector the PENELOPE software was used.

PENELOPE was developed by Salvat F., Fernández-Varea J. M. e Sempau J. in Fortran 77 and distributed by *Nuclear Energy Agency*. This software simulates the transport of electrons, positrons and photons in various geometries and materials. At the and, the software tracks the trajectory of every incident particle, with a related spectra. For electrons, the software analyze three events:

- elastic scattering;
- inelastic scattering;
- Bremsstrahlung emission.

For every process the cross section is given by associated tables or analytical results.

The utilized cross-sections are precise for the high part of simulations. For this reason, the software gives a good sampling of random particles interactions. Although, an high number of electrons and particles gives an high computational cost, because the high number of interactions of fast electrons. This problem can be solved using a mixed-system of simulations for particle tracking: *strong events*, characterized by polar deflection θ and W energy loss higher than default thresholds θ_c and W_c, are analyzed better; and *soft events*, where $\theta < \theta_c$ and $W < W_c$, are condensed. In this way the computation time is reduced rapidly if threshold values are increased. Threshold values are set by the user.
However, strong events determinate the trajectory of the particles. A mixed simulation is more convenient than a total condensed one, because it permits a better study of traces near interfaces and a reduction of imposed parameters[2].

Now a brief parenthesis of analyzed processes and the general algorithm of PENELOPE is presented, then results of this simulation are exposed.

B.1 Interaction modelling

B.1.1 Elastic scattering.

In an elastic scattering, initial and final quantum numbers of the target atoms are the same. In most of this cases the atom stays in a fundamental state.
Generally elastic scattering process determinate angular deviation of trajectory. We can note that the recoil of the target atom is approximately null, because the atom's mass ($\simeq 3600 m_e$) is much bigger than the mass of the incident electron.

Same as Chapter 1, the electron density $\rho_c(\vec{r})$ is calculated using Hartree-Fock approximation, assuming that if an atomic orbital is fully occupied, the distribution is spherical. Open shells are approximated using a mean distribution on the various direction. In this way the electron distribution of open shell is assumed spherical. The nuclei finite component of distribution, is obtained using the Fermi distribution:

$$\rho_n(r) = \frac{\rho_0}{exp[(r - R_n)(4ln3/t)] + 1}, \tag{B.1}$$

where $R_n = 1.07 \times 10^{-15} A^{1/3} m$, A the atomic mass and t is the surface's thickness. The surface's thickness is defined as the distance necessary to reduce ρ_n from 90 to 10% of his central value ($t = 2.4 \times 10^{-15} m$). The parameter ρ_0 is given by charge distribution normalization:

$$Z = 4\pi \int_0^\infty \rho_n(r) r^2 dr. \tag{B.2}$$

The electrostatic potential generated by the target is given by:

$$\begin{aligned} \phi(r) = {} & 4\pi e \left[\frac{1}{r} \int_0^r \rho_n(r') r'^2 dr' + \int_r^\infty \rho_n(r') r' dr' \right] \\ & -4\pi e \left[\frac{1}{r} \int_0^r \rho_e(r') r'^2 dr' + \int_r^\infty \rho_e(r') r' dr' \right]. \end{aligned} \tag{B.3}$$

If the field is static, the cross section can be obtained using quantum field theory principles, using the sum of the previous potential and the Hartree-Fock's exchange one. The final result is similar than that processes analyzed in the introduction chapter (see [19] and [1]).

B.1.2 Inelastic scattering.

Inelastic processes are the main cause of the energy loss by the electrons, in particular at low energies. For an amorphous material, for example atom or molecules without a privileged orientation, the differential cross section is independent by azimuthal angle. For this reason, the kickback energy Q is easier to use:

$$Q(Q + 2m_e c^2) = (cq)^2, \tag{B.4}$$

where q is the transferred Mandelstam's momentum. In Coulomb gauge and using Born approximation, the differential cross section of energy loss W is:

$$\frac{d^2\sigma_{in}}{dWdQ} = \frac{2\pi e^4}{m_e \beta^2 c^2} \left(\frac{2m_e c^2}{WQ(Q + 2m_e c^2)} + \frac{\beta^2 \sin^2\theta_r W 2m_e c^2}{[Q(Q + 2m_e c^2) - W^2]} \right) \frac{df(Q,W)}{dW}, \tag{B.5}$$

where θ_r is the angle between initial bullet momentum $\mathbf{p}$ and transferred momentum $\mathbf{q}$:

$$\sin^2\theta_r = 1 - \frac{W^2/\beta^2}{Q(Q + 2m_e c^2)} \left(1 + \frac{Q(Q + 2m_e c^2)}{2W(E + m_e c^2)} \right)^2. \tag{B.6}$$

The term $df(Q,W)/dW$ is the generalized force of the harmonic oscillator (GOS). GOS can determinate the effects of the inelastic scattering on the incident particle, according to the approximation. Analytic forms of GOS are available only for simplest interactions, than simulations are useful to determinate the property of GOS on Bethe's surfaces[1].

[1]The Bethe's surface are particular solutions of GOS on the plane (Q, W).

B.1.3 Bremsstrahlung emission.

This phenomena consist in a photon, with energy W, emitted by an electron, with an initial energy E, restrained or accelerated by an electrostatic field. The model of differential cross section for this process, related to an electron into a field generated by an atom with Z as atomic number and R shielding radius, is given by the Bethe-Heitler formula:

$$\frac{d\sigma_{br}}{dW} = r_e^2 \alpha Z(Z+\eta)\frac{1}{W}\left[\epsilon^2\phi_1(b) + \frac{4}{3}(1-\epsilon)\phi_2(b)\right], \tag{B.7}$$

where α is the fine structure constant, r_e is the classic electron radius and:

$$\begin{gathered}\epsilon = \tfrac{W}{E+m_e c^2} = \tfrac{W}{\gamma m_e c^2},\\ b = \tfrac{R m_e c}{2\gamma\hbar}\tfrac{\epsilon}{1-\epsilon},\\ \phi_1(b) = 4\ln(R m_e c/\hbar) + 2 - 2\ln(1+b^2) - 4b\arctan(1/b),\\ \phi_2(b) = 4\ln(R m_e c/\hbar) + \tfrac{7}{3} - 2\ln(1+b^2) - 6b\arctan(1/b)\\ -b^2[4 - 4b\arctan(1/b) - 3\ln(1+b^{-2})].\end{gathered}$$

The parameter η represent the radiation production in the atomic electrons' field. In the high energy limit $\eta \simeq 1,2$.

The equation B.7 is valid only if the electron energy and the energy of emitted photon are too much higher then the mass of the electron $m_e c^2$,but, fortunately, is useful in the highest part of the application of this phenomenon.

In the simulations, generally the energy loss W is sampled by the variable distributions, obtained by the integration of the cross section. In this way, angular distributions are lost, and they are must recovered with diverse approximations.

B.2 The algorithm

PENELOPE simulates the electronic transport, or *shower*, using various *subroutines*. The result is that a series of tracks characterized by segments of free-flights. In the end electrons interact with the material. The maximum step is set by the user for each material. In figure B.1 the flow chart of PENELOPE algorithm is shown [2].

Let focus on some subroutines, referring figure B.1 :

PEINIT configurations and characteristic of materials are read. Also, relevant scattering and energy quantities are evaluated and given by an output table;

GEOMIN reads geometry configuration, written in a new file by the user, evaluates volumes with quadrics and associates their material;

LOCATE finds the volume where the particle is positioned;

CLEANS initializes the stack of secondary particles, if they are present;

START forces the next event to be soft. This function is invoked when a new trace is created and when a particle walks through an interface;

JUMP determinate the trace's segment length until the next event. To simplify the computation, the function inserts ghost traces, which maintains the physical state of real particles;

[2]For a complete documentation of PENELOPE, see [19] and Appendix 1 of [2]

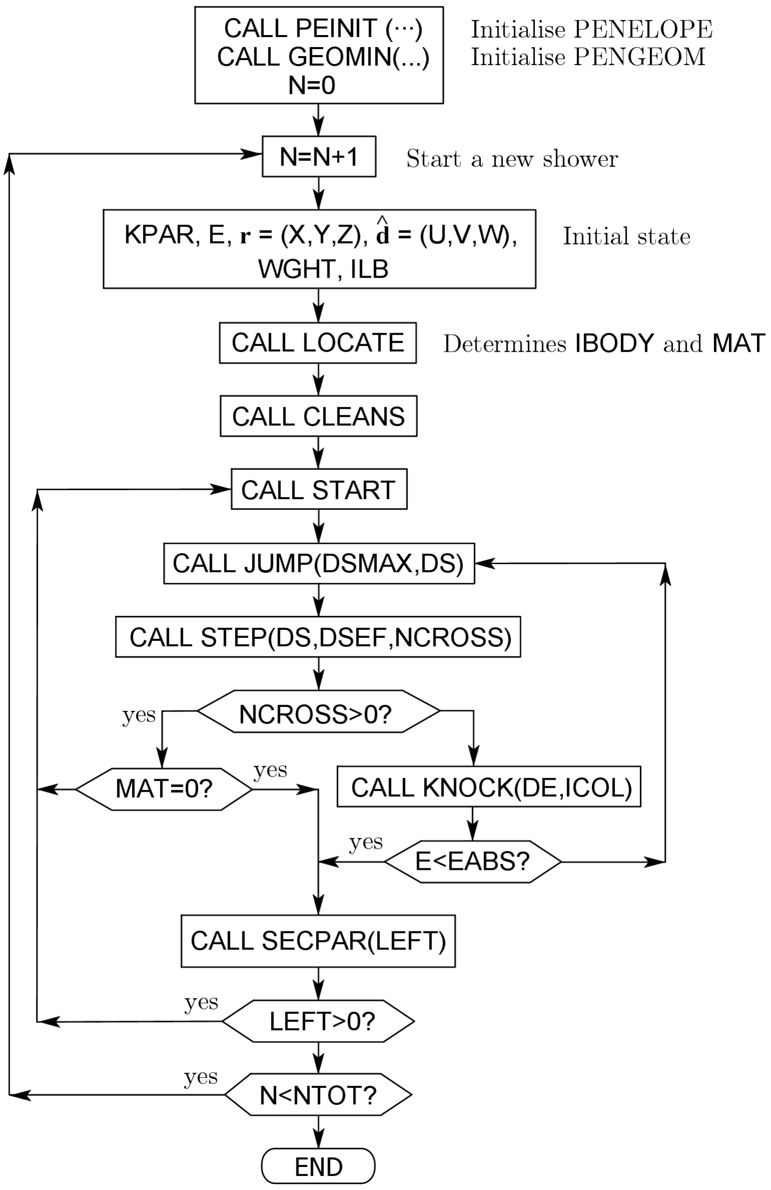


Figure B.1: PENELOPE *Flow Chart.*

STEP advances particles with distances calculated using the previous function, from (X,Y,Z) with the direction (U,V,W) in the laboratory rest-frame. If a particle walks trough a void region (MAT=0), it comes closer an other body present in the same region. If a particle walks trough a non void region, the particle is stopped.

KNOCK simulates an interaction and calculates new director cosins and the final energy. If presents, initial states of secondary particles are saved.

SECPAR set the initial state of eventual secondary particles and erase the respective stack.

For every interaction, the software verify that the average angular deflections and energy losses are higher than threshold values set by the user initially, to recognize soft and hard events.

Acknowledgement

This work have been carried out thanks to INFN-Laboratori Nazionali di Legnaro, Laboratory of Embedded Electronics of University of Trento, Fondazione Bruno Kessler in Trento and the *Conseil Européen pour la Recherche Nucléaire* (CERN) in Geneva.

I'm thankful with Professors Lucio Pancheri and Gian-Franco Dalla Betta and Andrea Ficorella for their support and their hints. Finally, I want to thank my colleague Tommaso Boschi for his simulation work with PENELOPE; and my supervisor, Professor Gianmaria Collazuol for the possibility and the help gaves me during my thesis work.

Padova, October 13th, 2016
Giuseppe Spadoni

Ringraziamenti

Di cose in questi ultimi due anni ne sono successe tante: Ho cambiato città, conoscenze, insomma tutto. O fortunatamente quasi.

"Quasi" perchè la mia famiglia - Mamma, Papà e i miei fratelli- mi hanno sostenuto come hanno sempre fatto, e in cambbio ho sempre cercato di dimostrare loro di crescere sempre di più, di essere diventato responsabile, di essere diventato grande.

"Quasi" anche perchè, anche se abbiamo cambiato tutti la nostra situazione, i miei amici più cari, provenienti ed aaffezionati alla mia stessa terra, sono ancora qui e anche loro cercano di realizzare i propri sogni. Per questo, come dice Ade, in un momento del genere bisogna ringraziare chi c'è e non deve accadere il contrario.
Io vi ringrazio e vi ringrazierò sempre per tutta la strada che abbiamo fatto insieme, per le risate, i litigi, le mangiate e i momenti di noia passati.

D'altra parte qui a Padova ho fatto nuove conoscenze e cose che, devo ammetterlo, al quale all'inizio ero restio.
Ho conosciuto tutti i miei amici con cui ho condiviso tutto, ma proprio tutto, praticamente a caso (a Random), grazie alla stupenda idea di Marta, che non ringrazierò mai abbastanza.
Ho conosciuto una miriade di persone diverse, da Ade e Matteo a Pietro e Cristian, da secchioni come me a festaioli, chi ha comunque preteso tanto da se stesso e non si è scoraggiato, facendosi in quattro, mettendoci la voglia e l'amore che può (sto parlando di voi, Alex, Bruna e Reida).

Ringrazio tutte le persone che hannno sopportato le mie lagne da scorbutico brontolone che sono, e purtroppo sempre sarò, e per via di una sfilza di gravissimi difetti e pochi pregi, anche se adesso non vogliono parlarmi, ma voglio che sappiano che comunque gli devo tanto.

Non riesco a dire altro che vi ringrazio tutti per quello che siete e per quello che avete fatto per me.

Di cose in questi ultimi due anni ne sono successe tante: Ho cambiato città, conoscenze, insomma tutto; ma sono stati due anni stupendi.

Padova mi mancherà tanto, cosi come i miei Padovani.
Grazie.

Padova, 13 Ottobre 2016
Giuseppe Spadoni

Bibliography

[1] MA Ali, Y-K Kim, W Hwang, NM Weinberger, and M Eugene Rudd. Electron-impact total ionization cross sections of silicon and germanium hydrides. *The Journal of chemical physics*, 106(23):9602–9608, 1997.

[2] Tommaso Boschi. *Studio della risposta di fotomoltiplicatori al silicio al passaggio di particelle cariche*. Universita' degli Studi di Padova., 2014.

[3] Robert D Chapman and Ronald JW Henry. Photoionization cross-sections for atoms and ions of aluminum, silicon, and argon. *The Astrophysical Journal*, 173:243, 1972.

[4] S Cova, M Ghioni, A Lacaita, C Samori, and F Zappa. Avalanche photodiodes and quenching circuits for single-photon detection. *Applied optics*, 35(12):1956–1976, 1996.

[5] S Cova, M Ghioni, A Lotito, I Rech, and F Zappa. Evolution and prospects for single-photon avalanche diodes and quenching circuits. *Journal of modern optics*, 51(9-10):1267–1288, 2004.

[6] Henri Dautet, Pierre Deschamps, Bruno Dion, Andrew D MacGregor, Darleene MacSween, Robert J McIntyre, Claude Trottier, and Paul P Webb. Photon counting techniques with silicon avalanche photodiodes. *Applied Optics*, 32(21):3894–3900, 1993.

[7] P.S. Marocchesi et.al. Development of an avalanche pixel sensor for tracking applications - proposal addendum. *Universita' degli Studi di Siena-INFN et. al*, 2014.

[8] Matthew W Fishburn. *Fundamentals of CMOS single-photon avalanche diodes*. fishburn, 2012.

[9] Majeed M Hayat, Unal Sakoglu, Oh-Hyun Kwon, Shuling Wang, Joe C Campbell, Bahaa EA Saleh, and Malvin C Teich. Breakdown probabilities for thin heterostructure avalanche photodiodes. *IEEE journal of quantum electronics*, 39(1):179–185, 2003.

[10] W Hwang, Y-K Kim, and M Eugene Rudd. New model for electron-impact ionization cross sections of molecules. *The Journal of Chemical Physics*, 104(8):2956–2966, 1996.

[11] G.Collazuol G.-F. Dalla Betta A.Ficorella P.S.Marocchesi F.Morsani L.Ratti A.Savoy-Navarro L.Pancheri, P.Brogi. First prototype of two-tier avalanche pixel sensors for particle detection. *Nuclear Instrument Methods in Physics Research A*, 2016. http://dx.doi.org/10.1016/j.nima.2016.06.094.

[12] Robert R Lucchese and Vincent McKoy. Accurate hartree-fock vibrational branching ratios in 3σg photoionisation of n2. *Journal of Physics B: Atomic and Molecular Physics*, 14(20):L629, 1981.

[13] Robert R Lucchese, Georges Raseev, and Vincent McKoy. Studies of differential and total photoionization cross sections of molecular nitrogen. *Physical Review A*, 25(5):2572, 1982.

[14] G.Collazuol G.-F. Dalla Betta A.Ficorella L.Lodola P.S. Marocchesi F.Morsani-L.Pancheri L.Ratti A.Savoy-Navarro M.Musacci, P.Brogi. Geiger-mode avalanche pixel in a 180nm hv cmos process for a dual-layer particle detector. 2016.

[15] Lucio Pancheri, David Stoppa, and Gian-Franco Dalla Betta. Characterization and modeling of breakdown probability in sub-micrometer cmos spads. *IEEE Journal of Selected Topics in Quantum Electronics*, 20(6):328–335, 2014.

[16] Ivan Rech, Antonino Ingargiola, Roberto Spinelli, Ivan Labanca, Stefano Marangoni, Massimo Ghioni, and Sergio Cova. Optical crosstalk in single photon avalanche diode arrays: a new complete model. *Optics express*, 16(12):8381–8394, 2008.

[17] Justin A Richardson, Lindsay A Grant, and Robert K Henderson. Low dark count single-photon avalanche diode structure compatible with standard nanometer scale cmos technology. *IEEE Photonics Technology Letters*, 21(14):1020–1022, 2009.

[18] Mohammad A Saleh, Majeed M Hayat, Paul P Sotirelis, Archie L Holmes, Joe C Campbell, Bahaa EA Saleh, and Malvin Carl Teich. Impact-ionization and noise characteristics of thin iii-v avalanche photodiodes. *IEEE Transactions on Electron Devices*, 48(12):2722–2731, 2001.

[19] Francesc Salvat, José M Fernández-Varea, and Josep Sempau. Penelope-2011: A code system for monte carlo simulation of electron and photon transport. 2011.

[20] V Saveliev and V Golovin. Silicon avalanche photodiodes on the base of metal-resistor-semiconductor (mrs) structures. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 442(1):223–229, 2000.

[21] Valeri Saveliev. Avalanche pixel sensors and related methods, September 18 2012. US Patent 8,269,181.

[22] R Van Overstraeten and H De Man. Measurement of the ionization rates in diffused silicon p-n junctions. *Solid-State Electronics*, 13(5):583–608, 1970.

[23] Leonardo Zanini. *Misura dell'efficienza di fotomoltiplicatori al Silicio per la rivelazione di fotoni nel lontano ultravioletto*. Universita' degli Studi di Padova., 2015.