Mauricio Montanares Sepúlveda
BSc Engineering

Design and implementation of a test environment for a MPPCs reader multi-channel ASIC

Submitted 12 August, 2021

Department of Electrical and Electronics Engineering , University of the Bío-Bío

Professors:

Krzysztof Herman, Angel Abusleme,
and Renzo Barraza

Abstract

The development of integrated circuits is a complex task. This task can be divided into three major areas: design, manufacture, and verification. In the context of Professor Ángel Abusleme's research, the master's student Renzo Barraza has designed and shipped to manufacture a chip that aims to read 4 SiPM (Silicon Photomultiplier, SiPM, Multi-Pixel Photon Counters, MPPC) in parallel. If the configuration of these detectors is correct, they can detect individual incident photons on their surface. Hoping the designed chip can be tested with real detectors in the future and then be used in particle physics experiments requiring the reading of photomultipliers of silicon (SiMP / MPPC). The chip manufacturer has already and now corresponds the process of characterizing it and checking its operation. That is the test and verification process of the post-silicon circuit (post-silicon validation).

Because there are currently no solution-specific tests environment for this ASIC (Application Specific Integrated Circuit); This work proposes the design and implementation of an embedded platform that will be allow the characterization and verification of the chip functionalities. This test platform should allow the chip and detectors to be configured correctly and also, this verification circuit should permit to control chip inputs and outputs when needed, as well as be able to record the data from these outputs.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Integrated Circuits

When I discovered the world of integrated circuits (IC), how these working and how they are made, this blew my mind, and it's not for the complexity of the design and fabrication process. It was the fact that these things work!

An integrated circuit consists of a single-crystal chip of silicon, containing both active and passive elements and their interconnection (Millman and Halkias, 1972).

These chips (and their internal elements) are mounted entirely on a semiconductor material. You can have both active and passive 'ingredients' on a single chip, but these days the number of active components (transistors) is the significant majority on an integrated device.

In the late 1950s was conceived the idea of placing multiple electronic devices on the same substrate. In 60 years, the technology has evolved from producing simple chips containing a handful of components to fabricating flash drives with one trillion transistors and microprocessors comprising several billion devices (Razavi, 2005).

CMOS (Complementary Metal-Oxide Semiconductor) technology is the base of these devices, and the design of these circuits depends on the type of

circuit. They are two big categories: digital circuits and analog circuits (and the mixture of these two called mixed signal circuits). Each of these groups has a different technic of design. Perhaps you also have two big categories of design: analog design and digital design (and the mixture of these two called mixed signal design).

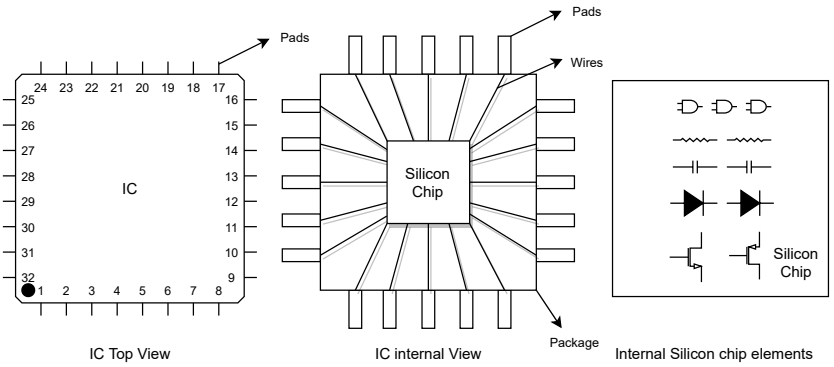


Figure 1.1: Integrated Circuit, different points of view

1.1.1 About the design of IC's

The proliferation of consumer electronics has been a driving factor in advancing integrated circuit (IC) design towards increasingly complex circuits and ever-smaller process technologies. The move towards design complexity has been aided by a mature and widely available set of Electronic Design Automation (EDA) tools in the digital domain. To take advantage of these tools circuit functions (e.g., signal processing) are implemented in the digital environment whenever possible (Galdames, 2021).

Analog and Digital integrated circuits (and also Mixed-signal) start from a top-down approach. Fig. 1.2 presented this graphically.

Three levels of abstraction can be readily identified during the design process (Galdames, 2021):

- The system-level, where system specifications are set, and functional blocks are identified.
- The circuit level, where circuit schematics are designed for each functional block.
- The layout level, where the circuit layout for all the functional blocks is designed, followed by floorplanning, placement, and global routing to generate the layout of the entire system.

After the design process, a next-level called *Post Silicon* arrives. At this level, the circuit, after manufactured, needs to be tested. The post-layout simulations are helpful to have an expected result in the post-silicon test. If this step fails, the IC is not ready for real-world applications, and the issues need to be identified (and corrected if possible). This Thesis concentrates on this stage and will be described a methodology for post-silicon verification for an ASIC(Application Specific Integrated Circuit).

1.1.2 About the fabrication of IC's

Fabricate an integrated circuit is complex and has many steps. This Thesis aims not to go deep into the Fabrication process of IC's; for that, we only will do a high-level overview of this process.

The fabrication of an Integrated Circuit (IC) it's a high-tech process. This process consists of a series of steps in which layers of the chip are defined through photo-lithography. The construction of an IC is based on layers of

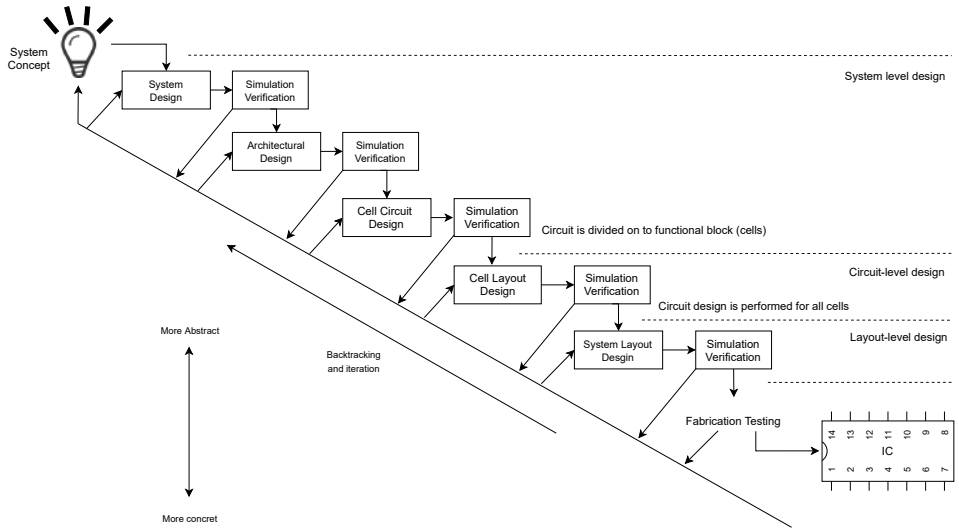


Figure 1.2: High-level view of the analog or mixed-signal design flow (Millman and Halkias, 1972; Galdames, 2021)

semiconductors, coppers, and other interconnected materials to form resistors, transistors, and other components. Transistors are fabricated on thin silicon wafers that serve as a mechanical support and an electrical common point called the substrate (Uyemura, 2012).For example, a cross-section view of a CMOS inverter is in Fig. 1.3. This image shows the different layers like n+ diffusion, p+ diffusion, polysilicon, metal, and silicon oxide. The fabrication consists of printed elements above the p-substrate(wafer) using different layers and interconnects these elements with metal.

In this diagram (Fig. 1.3), the inverter is built on a p-type substrate. The

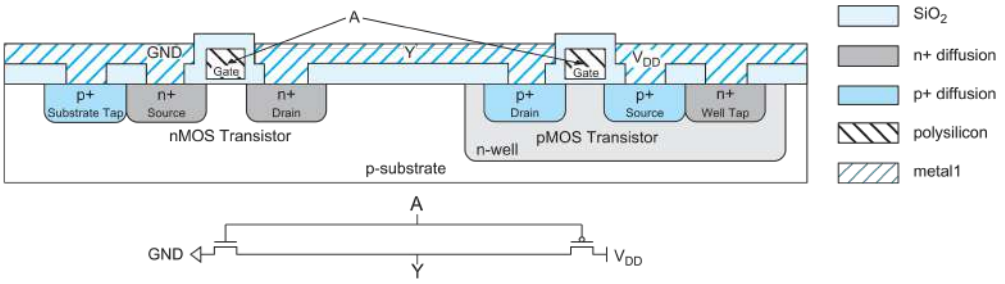


Figure 1.3: Inverter circuit. Cross-section view and schematic (Uyemura, 2012)

pMOS transistor requires an n-type body region, so an n-well is diffused into the substrate in its vicinity. The inverter could be defined by a hypothetical set of six masks: n-well, polysilicon, n+ diffusion, p+ diffusion, contacts, and metal.

Building another element (active or passive) can be done with the same process but with different steps. The lithography permits create a high level of complex IC's, and the cost is efficient. That is the reason why this process is the standard in the Silicon industry. (Razavi, 2005)

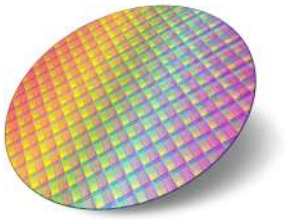


Figure 1.4: Silicon Wafer(p-substrate). Courtesy of WaferWorld

1.2 The verification process of IC's

In standard industrial practice today, analog, digital and mixed-signal circuits are still essentially verified based on repeated-simulations methodologies.

Verification is classified into two principal clusters depending on the moment when the verification is done: pre-silicon validation and post-silicon testing and test generation (Gielen et al., 2019) .

It is crucial to make a distinction between verification levels (pre-silicon and post-silicon):

- The pre-silicon design validation, which focuses on validating the correctness in the functionality of a design (Nahir et al., 2010).
- Post-silicon test generation, which focuses on verifying the correctness in the functioning of a fabricated (sub)system (Nahir et al., 2010).

1.2.1 Pre-silicon verification

If we examine Fig. 1.2 , it is possible to see different steps of verification. These stages are essential after every design step because it is crucial to detect all the possible problems after the fabrication process.

Simulation and verification steps are performed at each level to account for undesired effects (e.g., layout parasitics) and detect potential problems. If the design fails to meet specifications at some point in the design flow, redesign iterations are performed (Galdames, 2021).

Analog, Digital, and Mixed-signal circuit verification at design time can broadly be categorized into two approaches: formal methods and simulation-

based techniques. These approach are described bellow, and are based on (Gielen et al., 2019).

- Formal methods attempt to validate in some formal manner if the designed circuit is satisfying the correctness constraints. They typically step away from physics-based AMS (Analog/mixed-signal) simulations try to work at a higher level of circuit abstraction, as this can lead to a significant speed-up in design validation time, be it that they can suffer from inaccuracies due to this abstraction.

- Simulation-based techniques, on the other hand, start from the physics of the devices within the circuit and validate the design by sampling its performance through simulating the circuit several times under different inputs and conditions. This has the benefit that it takes many more physical details into account. However, since these simulations are time-intensive, this comes at the cost of an extensive validation time. In addition, the limited set of simulations carried out does not guarantee formally that the design is correct under all circumstances.

These days exist dedicated software for simulate and verified every step of the design process. The industry typically uses simulation-based validation as the default method.

This industry extended process has the benefit that the circuit is represented very reliably (up to the precision of the device models used and the parasitics incorporated in the netlist). However, the downside is that the simulation time can be excessively long, especially for transient simulations or simulating for time-dependent reliability (Gielen et al., 2019).

1.2.2 Post-silicon validation

Later the manufactured process is ready (Fig. 1.2), a new verifying stage is crucial to be sure of the correct functionality of the IC. This stage is denominated post-silicon validation. See the Fig. 1.5

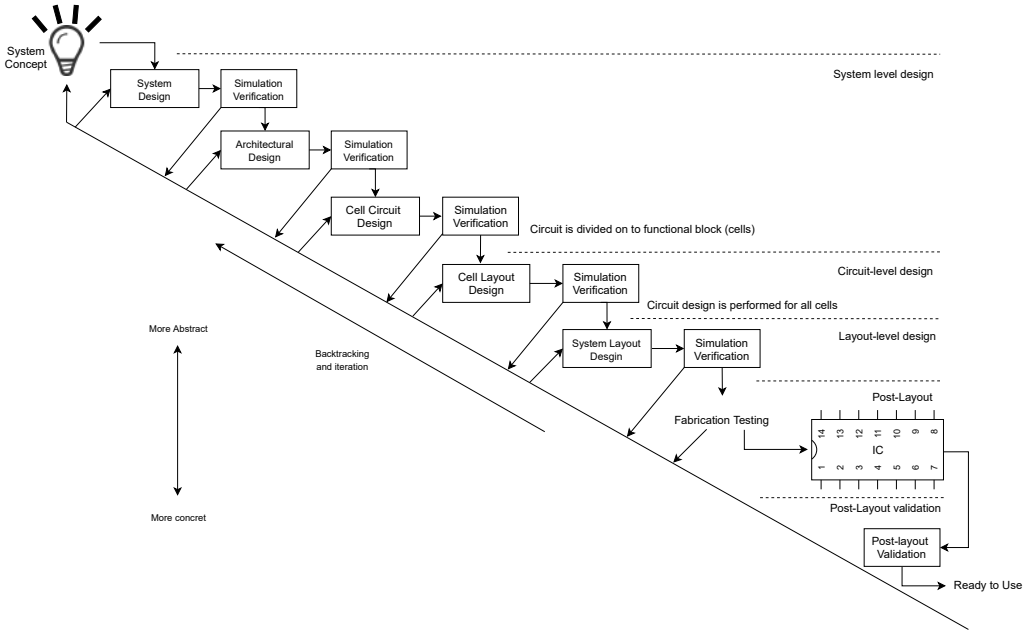


Figure 1.5: High-level view of the analog or mixed-signal design flow and Post-silicon validation stage context

Post-silicon validation involves operating one or more manufactured chips in actual application environments to validate correct behaviors over specified operating conditions. The objective is to ensure that no bugs escape to the field (Mitra et al., 2010).

Validation (post-silicon) has significant overlap with pre-silicon design

verification and manufacturing (or production) testing. Traditionally, most hardware design bugs are detected during pre-silicon verification and manufacturing defects are targeted by manufacturing testing.

While both manufacturing testing and pre-silicon verification continues to be essential. Post-silicon validation is becoming extremely necessary because of several unique aspects. These aspects are showed in Table 1.1 and are based on Mitra et al. (2010).

Table 1.1: Why post-silicon validation?

	Reasons why post-silicon validation is fundamental
1	We cannot rely on pre-silicon design verification alone to detect all design bugs. Simulation is several orders of magnitude slower than actual silicon. Formal verification is very useful for certain situations, such as the verification of individual arithmetic units or protocols, it faces scalability challenges for full chip-level verification. Hence, bugs that escape pre-silicon design verification are often detected during post-silicon validation.
2	In advanced technologies, several interactions between a design and the electrical state of a system are becoming significant, e.g., signal integrity (cross-talk and power-supply noise), thermal effects, and process variations. Such interactions can result in incorrect behaviors and are often referred to as electrical bugs. Accurate modeling of all these physical effects is usually very difficult during pre-silicon design verification.
3	Unlike manufacturing defects, post-silicon bugs may be caused by subtle interactions between a design and physical effects (the so-called electrical bugs) or by design errors (the so-called logic bugs). It may be very difficult to create accurate and effective fault models for such bugs.
4	Unlike manufacturing testing, where the primary objective is to detect defects, post-silicon validation involves localizing, root causing, and fixing bugs. Bug localization generally dominates post-silicon validation effort and costs.

Pre-silicon verification and post-silicon validation environments have fun-

damentally different characteristics.

Simulations at the pre-silicon levels are highly controllable, repeatable, observable, and hence easy to debug. On the other hand, it is maddeningly slow, limiting the depth and breadth of coverage that can be achieved even with farms of high-performance compute servers. Post-silicon validation is almost the inverse - extremely fast, but with very coarse controllability, uncertain reproducibility, very limited observability, and is very difficult to debug. The lack of controllability and observability makes it challenging to get meaningful coverage data with which to implement a coverage directed validation methodology (Nahir et al., 2010).

These differences are recapitulated and exhibit in Table 1.2. It should also be considered that post-silicon validation also suffers from platform stability and electrical and circuit issues, increasing the problem.

Table 1.2: Pre-silicon vs Post-silicon

Pre-silicon verification	**Post-silicon validation**
Observable	Very limited observability
Highly controllable	Coarse controllability
Repeatable	Problematic reproducibility
Easy to debug	Difficult to debug
Slow(simulation)	Extremely fast

While many methods and tools have been developed and are widely used in industry to both formally validate and automatically generate tests (ATPG) for digital subsystems, the AMS(Analog/mixed-signal) verification tools are lagging (Guo et al., 2015).

This Thesis is focused on an analog/mixed-signal ASIC and its verification. The inherent obstacles of verifying analog circuits lie in the facts that

(Gielen et al., 2019; Nahir et al., 2010):

- There is no practical or commonly used method to formally describe their function, whereas commercial simulators are used extensively in all design steps.
- The performance of analog circuits is parametrically sensitive to all kinds of inequalities (process, supply, temperature, aging), which makes their verification complex.
- The influence of fabrication defects on analog circuit functionality is less predictable.

Pre-silicon tools give better control over the corners we want to hit. This control is not there in post-silicon tools. Presently it is tough to measure our coverage in post-silicon validation. Moreover, these methods are based on the assumption that we know all the paths we want to cover in the hardware, which we do not. We require developing and implementing the stimuli available in pre-silicon for the post-silicon experiments (Nahir et al., 2010).

It is seldom to see a unified plan for verification and validation (especially on AMS). We need to act on specifics parts of the circuit that need to be tested. When and how to do it.

We can propose an easy and high-level view example: Suppose a chip, this device was stimulated in post-layout simulations with some known inputs, and the chip outs were collected. These data were checked for the designer, and the behavior of the circuit accomplishment the design specifications. So the circuit will be fabricated.

Once the chip fabrication is ready, it's the moment to do a Post-silicon test. Recreated(in real life) the original stimulus (used in post-layout simulation) and use these signals for stimulus the chip inputs. The outs(in Post-silicon test) can be measured and compared with the data collected from the releases in the post-layout simulation. See the Fig. 1.6

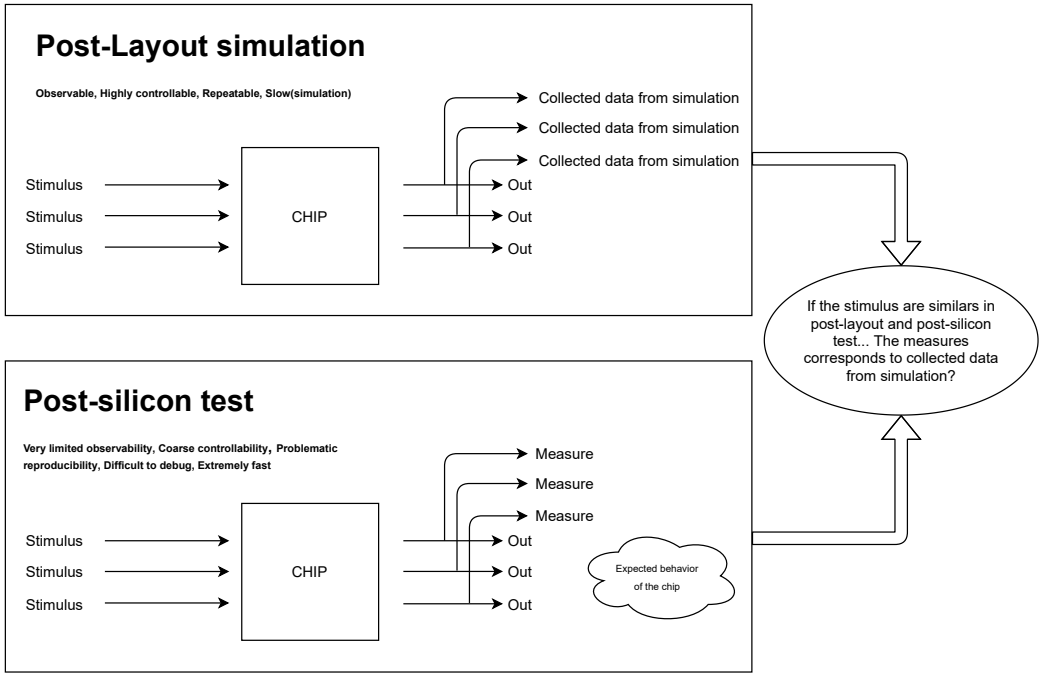


Figure 1.6: High-level example view of Post-silicon test

1.3 About the Silicon Photomultipliers

Silicon photomultipliers (Silicon Photomultiplier, SiPM, Multi-Pixel Photon Counters, MPPC) are detectors consisting of multiple p-n silicon junctions that, when polarized with voltages close to their breakdown voltage, can detect individual photons incident on their surface (Altamirano, 2021). As stated in Sadygov et al. (2006), a silicon photomultiplier is a monolithic

solid-state photodetector composed of an array of hundreds or thousands of photodiodes put in parallel. These are independent of each other and are connected to a common read node (fast output).

The primary elements of the SiMP are called pixels(photodiode, see Fig. 1.7b). In the case of the SiMP model "MICROFCSMTPA-10010-GEVB" (ON (2021)), this device is formed of a large number (hundreds or thousands) of microcells. Each microcell is an avalanche photodiode with its own quench resistor and a capacitively coupled fast output. Fig. 1.7a.

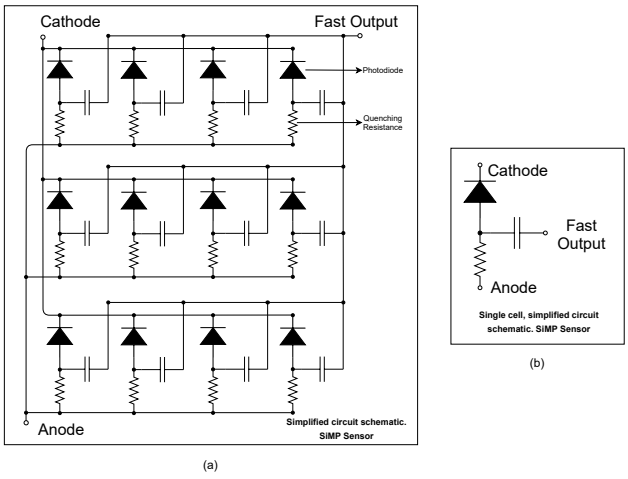


Figure 1.7: (a) Simplified circuit schematic of SiMP. (b) Single cell, simplified circuit schematic. SiMP Sensor. (ON (2021)).

1.3.1 Operation mode

When a photon arrives at the detector, activate some pixels, and if a pixel is activated (for an incident photon), the pixel produces a signal out. Each pixel of the MPPC produces the same out amplitude when a photon is detected. A representation of this is shown in Fig. 1.8.

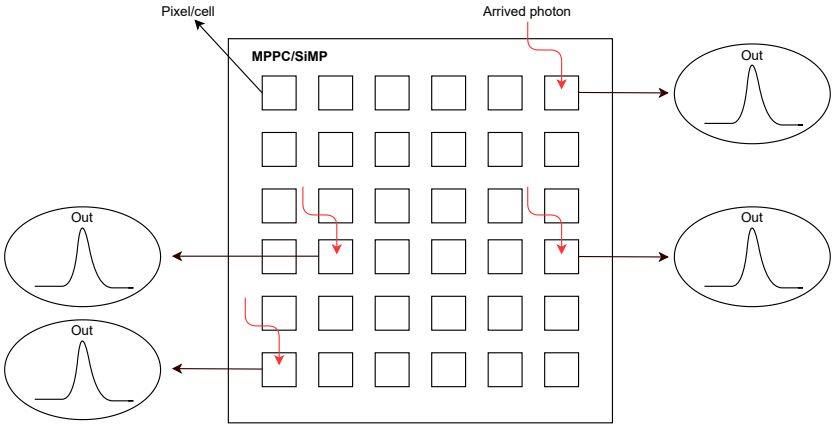


Figure 1.8: A high-level view of MPPC, arrived photons and pixel's outs (HAM, 2021).

And what happens if more than one photon arrives at the same time? In this scenario, the pixel's output signal will be the same as that for one photon, but the amplitude of the MPPC output will be the sum of the individual cell signals.

Fig. 1.9 shows output pulses from the MPPC obtained when it was illuminated with the pulsed light at photon counting levels and then amplified with a linear amplifier and observed on an oscilloscope. As can be seen from the figure, the pulses are separated from each other according to the number of detected photons such as one, two, etc (HAM, 2021). Measuring the height of each pulse allows estimating the number of detected photons.

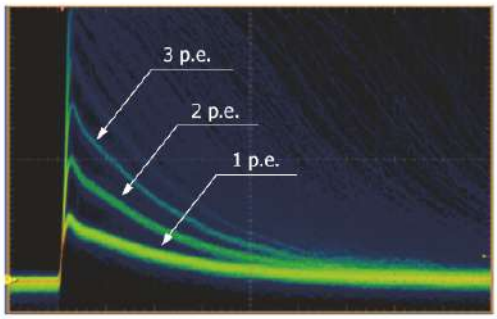


Figure 1.9: Pulse waveforms when using a linear amplifier (HAM, 2021).

2 A short introduction to the custom ASIC for reading SiMP detectors

The chip is a multichannel ASIC and has four analog channels with four outputs each. Of these four outputs, 3 are analog signals, and one is digital.

Renzo Barraza has designed a chip that aims to read 4 SiPM (Silicon Photomultiplier, SiPM, Multi-Pixel Photon Counters, MPPC) in parallel. If the configuration of these detectors is correct, they can detect individual incident photons on their surface.

2.1 Description and features of the ASIC

This IC can read 4 SiMP detectors in parallel, process these measurements internally, and through 4 outputs (3 analog and one digital), deliver information on the number of photons that arrived at each detector. The number of photons that arrived at the detectors must be extracted from the analog and digital signals (chip outputs). These estimates and their methods are described in detail in section 3.2.

Some relevant characteristics of the ASIC:

- The chip has four analog channels for 4 MPPC (SiMP) detectors.

- Each analog channel has an 8-bit DAC, which allows the user of the readout circuit to fine-tune the gains of the SiPM by adjusting its bias voltage, in addition to compensating for the dark noise mentioned in chapter (Altamirano, 2021) . Due to the limitations of CMOS technology, the maximum voltage that can be delivered is close to 1.8V (relative to ground). This limitation prevents the Chip itself from being able to directly bias the MPPC detectors.

Fig. 2.1 is a graphical summary of these points.

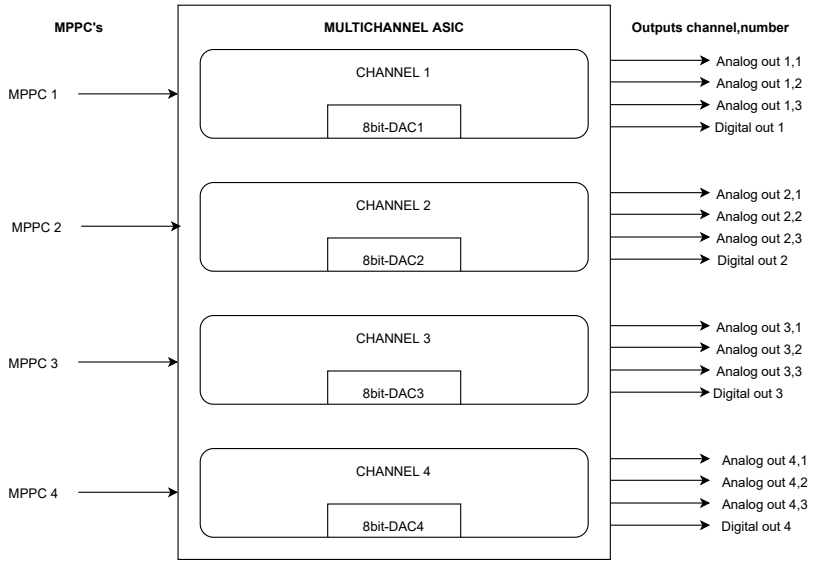


Figure 2.1: ASIC high-level overview

2.2 Inputs and on-chip signal processing

The chip has four analog inputs and four outs, three analogs and one digital output. In details:

This ASIC has four channels, and each channel has one analog input. Those inputs are connected to the MPPC's, and when photons arrive at the detectors, a flow of charge is produced and the chip reads this current using a specific circuit.

The reading circuit for the SiMP detectors corresponds to an open-loop transimpedance amplifier (TIA) type circuit, which the chip designer justifies based on the objectives proposed in the thesis presented by R.Barraza (Altamirano, 2021).

A high-level circuit representation of the above is shown in Fig. 2.2, where "-HV" is a high negative voltage, and it is used to polarize the detectors.

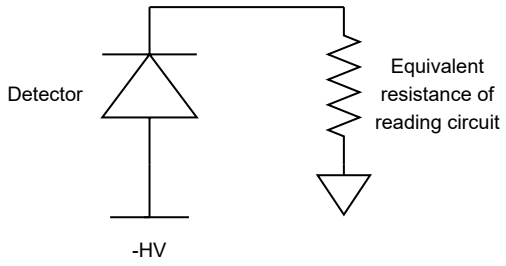


Figure 2.2: Read out circuit representation (Altamirano, 2021)

Employing the TIA is possible to read the flow of charge (coming from the detector, drift current), and after processing the voltage output signal of the TIA (Fig. 2.3). Using the voltage output is possible to obtain information about how many photons arrived at the detector. The methods for doing that are described in section 3.2

2.2.1 Post-TIA Processing

Once the TIA has read the signal, two things must be measured:

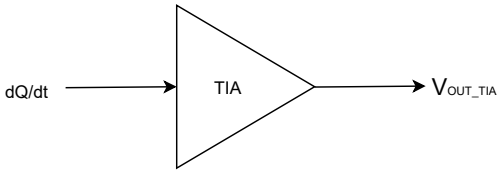


Figure 2.3: Transimpedance Amplifier (TIA)

1. The instant the charge is received.

2. Number of pixels activated in the detector.

To know the first one, the designer of the chip decided to implement a discriminator circuit. This circuit is activated when the voltage signal produced by the TIA indicates that one or more pixels have been triggered/activated in the detector. The output of the discriminator, therefore, delivers a digital signal (on/off transition) and this signal lasts as long as the MPPC continues to provide current. The information supplied by the discriminator can be used to know the number of pixels activated in the detector. Additionally, the designer decided to implement an extra circuit block to improve the results. This new circuit is an integrator circuit, and its purpose is to integrate the output voltage signal of the TIA. The information that provides this block can be added to that already delivered by the discriminator to read more accurately the number of activated pixels (Altamirano, 2021).

A representation of the internal component of each of the four channels is shown in Fig. 2.4 and shows that the signal produced by the SiMP connects to the VIN_TIA_EXT node. The potential of this node can be altered due the voltage produced by the DAC of the channel(four channels, and each channel has one DAC). This DAC is capable of doing a fine-adjusting of the SiPM bias voltage (with the limitations of CMOS technology explained in

section 2.1)

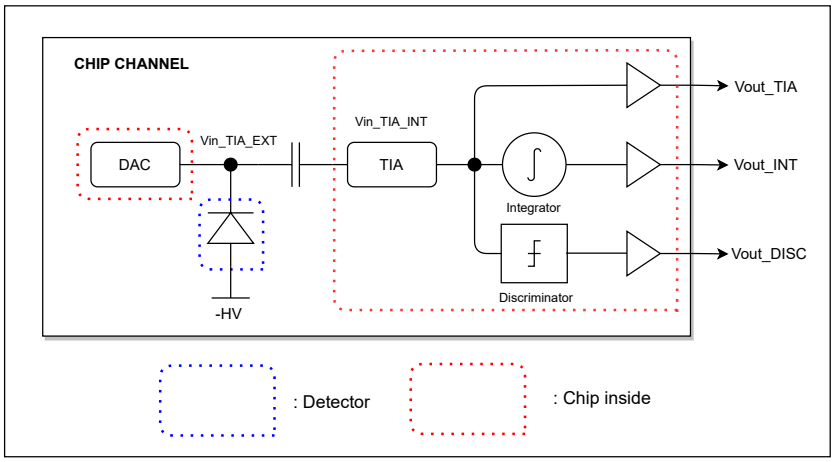


Figure 2.4: Internal components of each of the four channels in the ASIC (Altamirano, 2021).

2.3 Description of the chip outputs

The chip for each channel has four outputs:

- VOUT_INT_DER and VOUT_INT_IZQ: These are the differential outputs of the integrator (VOUT_INT in the figure). This output is used to identify few triggered pixels.

- VOUT_TIA: Direct output of the TIA. It is employed to identify a number greater than hundreds of triggered pixels.

- VOUT_DISC: It is the output of the discriminator. It serves to identify the moment when a signal arrives from the SiPM and can also be used to identify a neighborhood bounded by triggered pixels.

These points are based on the description of the outputs gives by the designer of the ASIC in (Altamirano, 2021). In Fig. 2.4, it is possible to observe the outputs of one channel.

3

Post-layout simulations and estimation methodologies for counting photons using the ASIC

In Fig. 1.2, the last verification step is called post-layout verification. At this stage, the designer did appropriate simulations based on the problem specification described on chapter 2. These simulations were used to estimate the number of photons arrived at the detector and the time of this event. For this, the designer of the chip used methods that will be described in this chapter.

3.1 Post layout simulations

Using software and some techniques described in section 1.2, the designer simulates some conditions for validating the behavior of his circuit. The ASIC is designed to be capable of two main things: estimate the number of photons arrived at the detector and the time of this event.

To do these two things, simulated the arrival of photons is necessary. So, how to do that?

3.1.1 Single-photon arrival simulation

The author of the circuit used as an input signal a current pulse of $0.1ns$ with an amplitude of $4mA$. This, because the electrical charge delivered by said current pulse is equivalent to the charge delivered by 1 pixel triggered in the *Hamamatsu S14160-3050HS* detector.

The aforementioned current pulse was used for stimulating the channels of the ASIC. The response of the circuit and the outs are exhibited in the figures 3.1, 3.2 and 3.3.

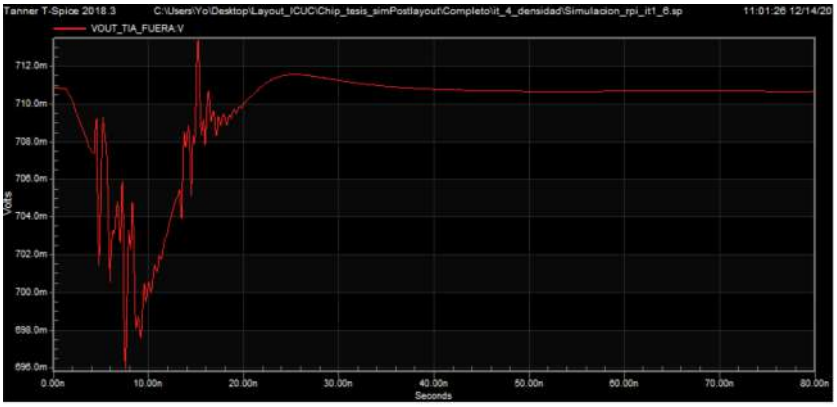


Figure 3.1: TIA Out for one pixel activated (Altamirano, 2021)

3.1.2 Multi-photon arrival simulation

The current pulse used for simulating the arrival of one photon is linearly scaled for simulating more arrival photons (more pixels activated). This linearly scaled affects just the signal's amplitude, but the duration of this signal is equal for every quantity of simulated photons (Table 3.1).

For example, to simulate the activation of two pixels of the detector (MPPC/SiMP), the current pulse will be a pulse of $0.1ns$ with an amplitude of

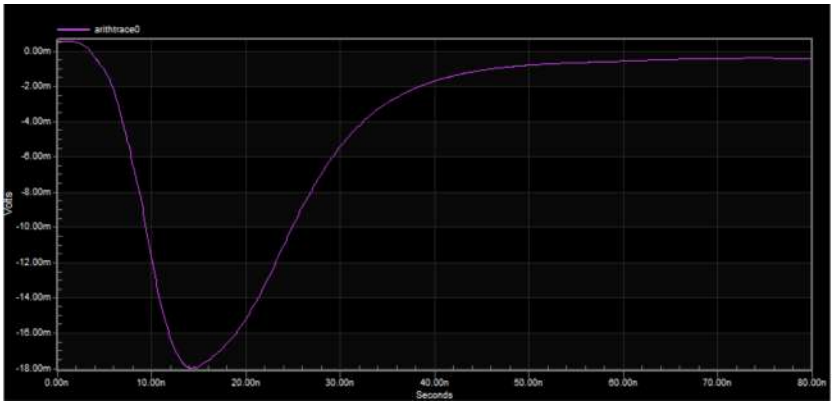


Figure 3.2: Integrator Out fo one pixel activated (Altamirano, 2021)

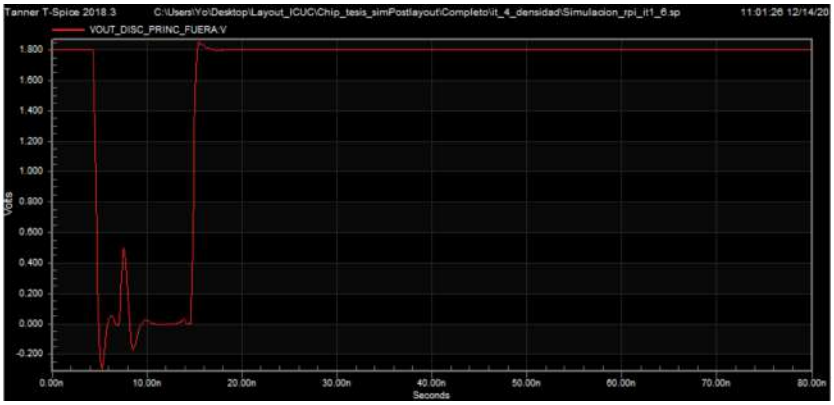


Figure 3.3: Discriminator Out for one pixel activated (Altamirano, 2021)

$8mA$. In the case of simulating the activation of four pixels, the current pulse will be a pulse of $0.1ns$ with an amplitude of $16mA$. And so on. Table 3.1, Fig 3.4 and Fig 3.5 summarize this.

The third column of Table 3.1 shows the equivalent charge for a pulse of $0.1ns$ with this correspondence amplitude. That data will be necessary for the next chapters.

Generate a pulse whit the values of amplitude and duration of $0.1ns$ can be challenging. Nevertheless, it is possible to produce a pulse that the

Table 3.1: Pixels, Current and Charge relation

Pixels	Current[mA]	Charge[q]
1	4	4E-13
2	8	8E-13
3	12	1.2E-12
4	16	1.6E-12
5	20	2E-12
6	24	2.4E-12
7	28	2.8E-12
8	32	3.2E-12
9	36	3.6E-12
10	40	4E-12
...	...	...
1000	4000	4E-10
...	...	...
2000	800	8E-10
...	...	...
3000	12000	1.2E-09

integration area will be equivalent to the equivalent charge for n number of pixels, following the current equation 3.1.

$$\frac{dQ}{dt} = C\frac{dv}{dt} \tag{3.1}$$

3.2 Chip signal reading. How to read/interpret the analog outputs of the chip?

On Altamirano (2021), two methodologies are mentioned to measure the outputs and estimate the number of activated pixels. We will refer to these as:

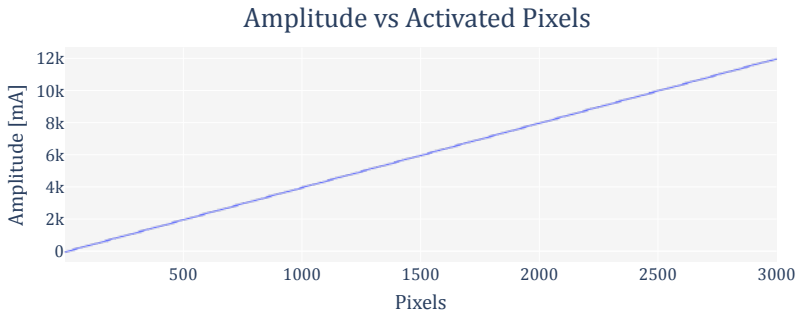


Figure 3.4: Current Amplitude vs activated pixels.

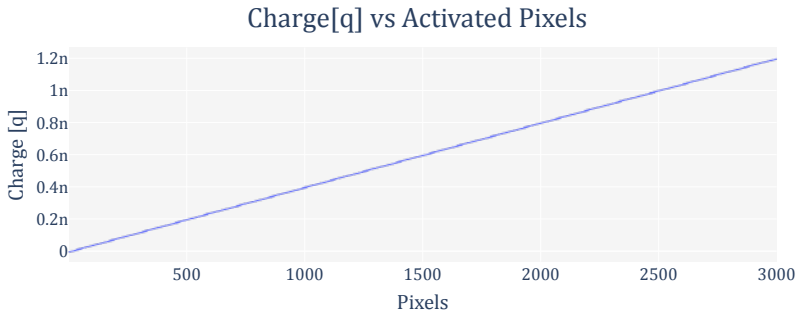


Figure 3.5: Charge vs activated pixels.

1. Peaking time approach

2. Maximum amplitude approach

3.2.1 Peaking time approach

This approach has two stages: calibration and measurement. Calibration is done with just one pixel activated ($N = 1$). At this point, a time difference Δt is measured between the moment of the discriminator trigger ($Vout\_DISC$ transition on / off) and the moment of maximum amplitude (*peaking time*) of the $Vout\_INT$ output for one activated pixel. The next measurements (when N increase) will be performed with the same *peaking time* measured

in the calibration stage ($N = 1$). It is essential to mention that for a small N, therefore a low amplitude input signal, this method works well, but as N increases, the transistors leave their region of linearity, which will cause imprecision in the measurement of the valid maximum of the *Vout_INT* signal Altamirano (2021). This approach can be seen in the following Fig 3.6.

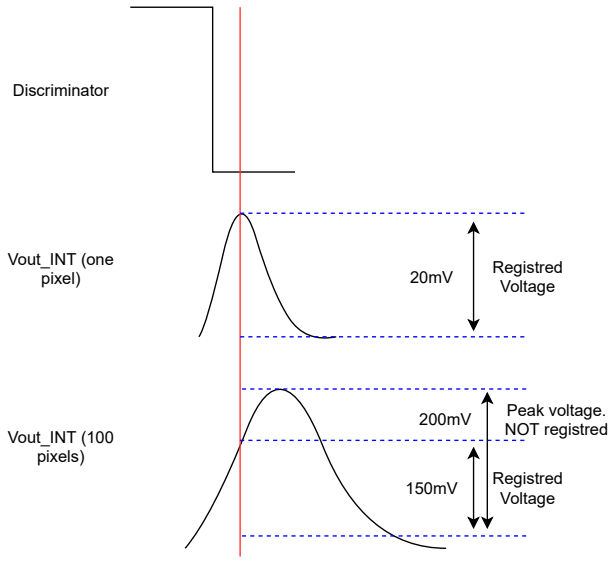


Figure 3.6: Peaking time approach (Altamirano, 2021). Voltages on the figure are just referential examples.

3.2.2 Maximum amplitude approach

The second approach is to assume that the user will identify the maximum amplitude of the analog outputs. Therefore the maximum value of said signals is recorded regardless of when this occurs; This approach can be seen in the following Fig 3.7.

In addition to these two methods, it is possible to use the digital signal

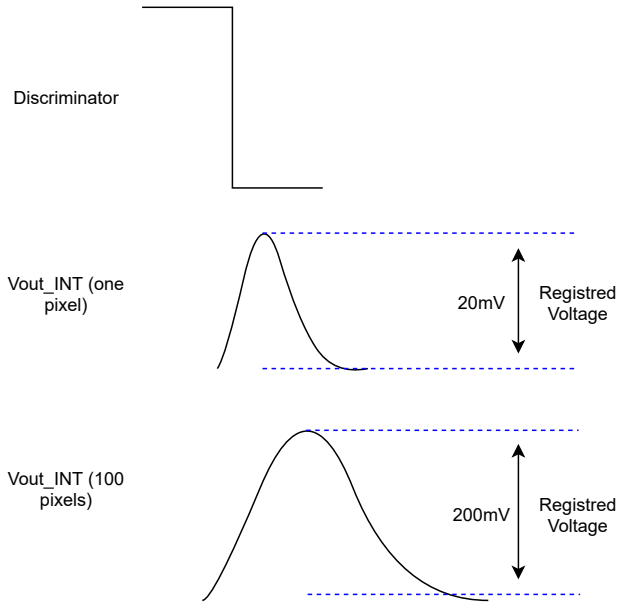


Figure 3.7: Maximum amplitude approach (Altamirano, 2021). Voltages on the figure are just referential examples.

from the discriminator and, depending on the duration of the pulse, estimate the number of pixels triggered using the *Time Over Threshold* method. This method is described in Orita et al.; Bagliesi et al. (2011); Altamirano (2021) and is used in circuits of similar applications. The method consists of relating the duration of the digital pulse with a certain number of activated pixels.

3.2.3 Performance of methodologies

The following Figs 3.8, 3.9 shows the performance of both methodologies when measuring the outputs of the TIA and integrator. Additionally, the duration of the discriminator pulses to identify the number of pixels it is showed in Table 3.2.

The chip designer mentions that based on their results, it can be stated

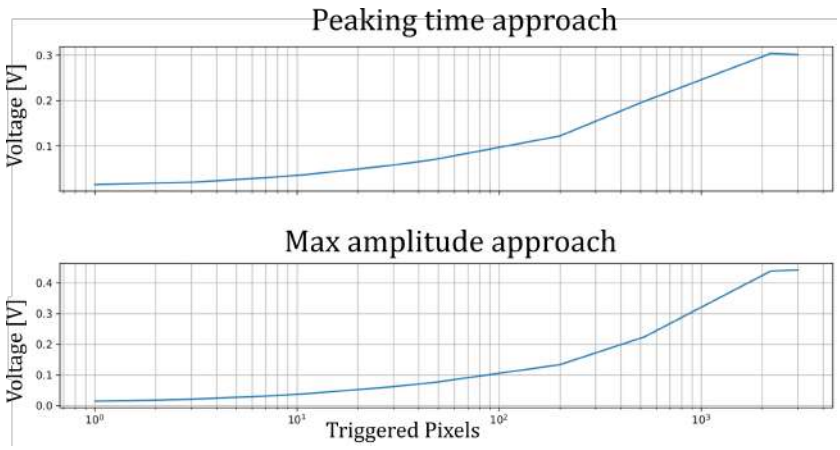


Figure 3.8: Results of the approach. TIA out (Altamirano, 2021).

that: Both measurement approaches of the integrator cannot resolve differences of 1 triggered pixel, but it can resolve differences of 2 triggered pixels. This capacity degrades as more amplitude signals are received from the SiPM. A summary of the ability to resolve multiple pixels using the various analog outputs is provided below and it is summarized on Tables 3.3 and 3.4.

- It should be noted that the TIA data are not reliable with few pixels because there the oscillations caused by the digital buffer abnormally. It is worth recalling that TIA output is intended to deliver information of hundreds or more pixels.

- Based on the data mentioned above, using the *Time Over Threshold* technique to resolve signals of few pixels becomes necessary.

Table 3.2: Duration of discriminator pulses according to number of pixels triggered in the detector

Number of activated pixels	Discriminator pulse duration [ns]
1	11.4
2	17.68
3	21.24
7	31.65
11	38.07
29	54.88
47	65.88
199	>100

Table 3.3: Ability to resolve pixels of the TIA.

Number of triggered pixels in the detector	**Resolution capacity of pixels measuring maximum of the signal(max amplitude approach)**	**Pixel resolution capacity measuring at the moment of maximum signal for 1 pixel (peaking time approach)**
1 - 11	2	2
11 - 29	3	3
29 - 47	10	6
47 - 199	10	20
199 - 521	35	100
521 - 2207	250	Not distinguished
2207 - 3000	Not distinguished	Not distinguished

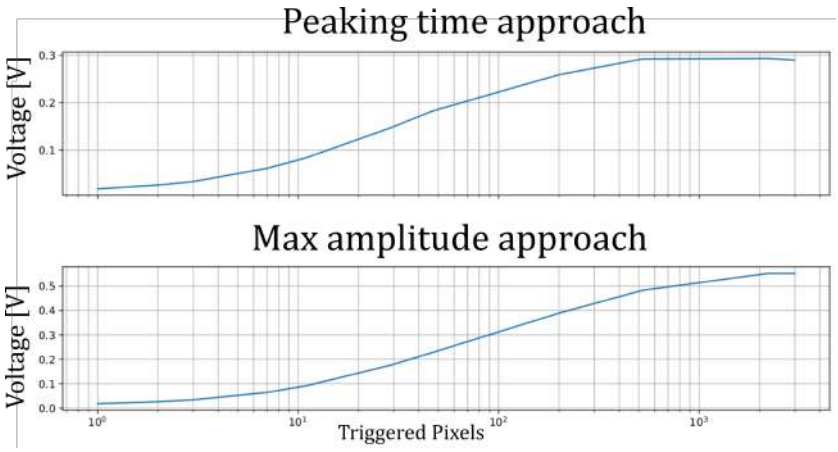


Figure 3.9: Results of the approach. Integrator out (Altamirano, 2021).

Table 3.4: Ability to resolve pixels of the Integrator.

Number of triggered pixels in the detector	**Resolution capacity of pixels measuring maximum of the signal(max amplitude approach)**	**Pixel resolution capacity measuring at the moment of maximum signal for 1 pixel (peaking time approach)**
1 - 2	1	1
2 - 3	1	2
3 - 7	1	1
47 - 199	2	2
11 - 29	3	3
29 - 47	4	4
47 - 199	7	8
199 - 521	9	11
521 - 2207	20	40
2207 - 3000	575	Not distinguished

4

Proposed environment for testing the chip

The process of validation and characterization of the chip necessarily requires a suitable test environment. This test environment must have the capabilities to properly condition the circuit to stimulate the inputs and analysis of the outputs. Additionally, the test environment must provide stimulation signals and do a proper registration of the outputs. With the recorded information of the outputs and using the estimation methods described in section 3.2, estimating the number of triggered pixels will be possible. In the practice, this environment will be an specific board with all the necessary elements for the characterization and validation process.

In previous works, Orita et al.; Bagliesi et al. (2011); Galdames (2021) a circuit is designed and printed on a plate, which allows the IC to be correctly conditioned to proceed with the stimulation of the inputs and analysis of the outputs. The ASIC designer proposes as future work the design of a PCB (printed circuit board) to validate and test the ASIC.

The design of this characterization and validation environment is set out below. Details related to the implementation of the methods will be presented in this chapter.

The validation environment must have:

- A negative HV(high voltage) generator for the correct polarization of

the detectors.

- Current pulse generator for stimulating the inputs.
- Mechanisms for the control and reading of analog and digital data that allow the implementation of the pixel estimation methodologies described in the section.

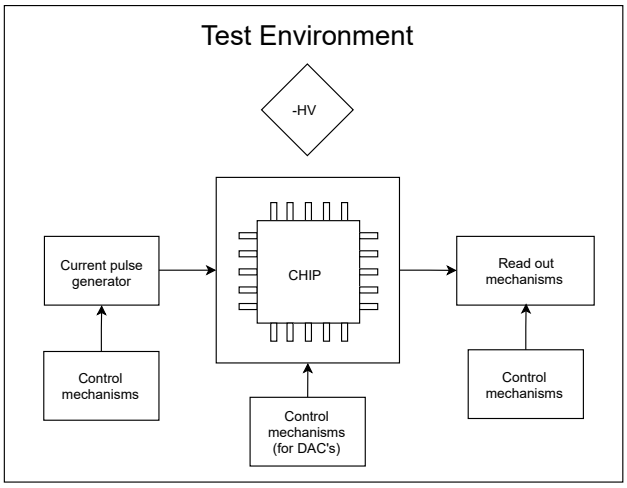


Figure 4.1: Block diagram and high-level view of the test and characterization environment.

4.1 Negative HV generator

A negative HV generator it is for the correct polarization of the detectors. The specifications of the *"MICROFCSMTPA-10010-GEVB"* detectors indicate a reverse bias voltage close to -24V. The PCB must supply this voltage. Based on the specifications of the detector datasheet, this high-negative voltage must be generated by a Charge Pump type circuit.

The Charge Pump (CP) is an electronic circuit that converts the supply voltage VDD to a DC output voltage Vout that is several times higher than VDD. Unlike the other traditional $DC - DC$ converters, which employ inductors, CP are only made of capacitors and switches (or diodes) [1]. CP under specific conditions can also produce a voltage lower than the input voltage, generating a negative voltage.

They are different topologies of charge pump circuits, and the majority of these are based on multiplication stages. The idea behind the behavior of these types of power sources is to use semiconductor elements (diodes or transistors) combined with two complementary control signals to guide the direction of the current connecting and disconnecting different stages of the circuit to produce an increment (or decrements) of the output voltage.

For example, on Tanzawa and Tanaka (1997) the authors analyze the dynamics of a Dickson CP circuit. The topology of the circuit doubler voltage is presented in Fig 4.2.

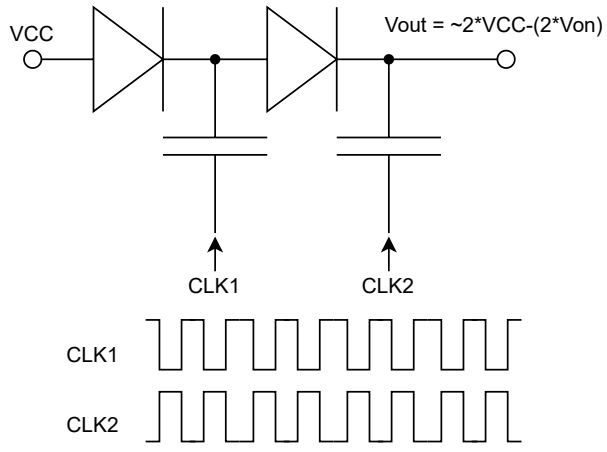


Figure 4.2: Dickson CP, voltage doubler and complementary clock (digital) signals (Tanzawa and Tanaka, 1997).

To improve the results of this topology is recommended to use *Schottky diodes* because the parameter V_{on} is smaller than other diodes(Palumbo and Pappalardo, 2010). It is also possible to use transistors instead of diodes, this is a very popular practice to implement these CP sources in integrated circuits.

For example a Dickson charge pump with MOSFET implementation is shown in Fig. 4.3. It is a typical n-stage Dickson charge pump. The input $CLK1$ and $CLK2$ are two out-of-phase clocks with amplitude V_{clk} . These two clocks will increase the potential voltage in capacitors by transferring charges in the capacitor chains through diode-connected MOS transistors. The coupling capacitors will be charged and discharged during each half clock cycle (Lin, 2012a).

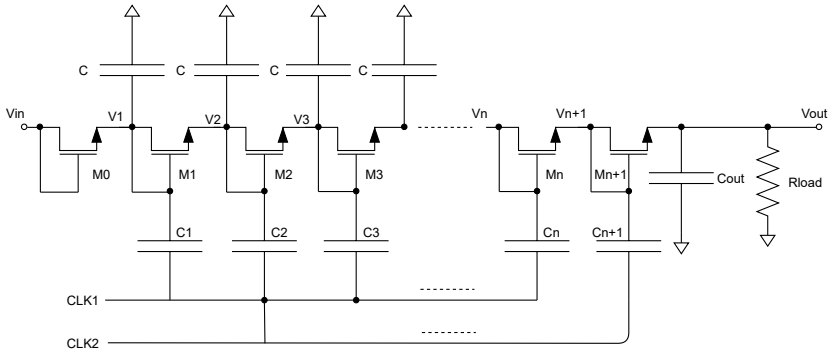


Figure 4.3: A typical n-stage Dickson charge pump (Lin, 2012a).

On equation 4.1 voltage V_t is related to the node voltage of each stage and the voltage difference between the voltages of the nth stage and $(n+1)th$ nodes is given by:

$$\Delta V = V_{n+1} - V_n = V'_{clk} + V_t \tag{4.1}$$

On 4.2 C_s as the stray capacitance at each node and C are the coupling capacitances. The voltage gain V'_{clk} can be express as:

$$V'_{clk} = \left(\frac{C}{C + C_s}\right) V_{clk} \tag{4.2}$$

For an N stage charge pump, the voltage at the output node is as follow:

$$V_{out} = V_{DD} + N(V'_{clk} - V_t) - V_t \tag{4.3}$$

The equation above is the ideal output voltage when there is no output charge and no output current delivered. When the charge pump is connected to an output load, the load current at a clock frequency f , is given by (Lin (2012a)):

$$I_{out} = f(C + Cs)V_L \tag{4.4}$$

V_L is the voltage drop per stage for supplying the load current. Therefore, the output voltage will be reduced an $N \cdot V_L$ voltage. So the output voltage with load will be rewrite as follow (Lin (2012a)):

$$V_{out} = V_{DD} + N\left(\frac{C}{C + C_s}\right) V_{clk} - V_t - \frac{I_{out}}{f(C + Cs)} - V_t \tag{4.5}$$

Equations 4.1 ,4.2 , 4.3, 4.4 and 4.5 are based on Lin (2012a)

This type of circuit is easy to find as integrated circuits, where manufacturers guarantee certain output values for specific conditioning parameters. For example, the LTC3261 is a high voltage inverting charge pump that operates over a wide $4.5V$ to $32V$ input range and can deliver up to $100mA$ of output current, and the output voltage will be $-V_{in}$. Based on the datasheet,

this IC is a good option for implement the HV negative source (Lin, 2012b).

4.2 Current pulse generator for stimulating the inputs

Before using real detectors, the outputs of these sensors must be emulated in such a way as to have control over the timing of the current pulse triggering. Having control over the pulse triggering moment is essential to perform the characterization and validation of the ASIC. To achieve this, a charge generator circuit will be required.

The current signals used in the post layout simulations (Chapter 3) have amplitudes ranging from $4mA$ to $12A$, with a duration of $0.1nsfor$ any of the amplitude values in Table 3.1. Current pulses of the same order of magnitude as the signals used in the post-layout simulations (ns) must be generated for testing purposes.

The chip is designed to read charge(and the current associated with this flow of charge), so the fundamental objective will be to generate current pulses where the delivered charge can be related to N of activated pixels using the linear relationship mentioned in section 3.1.2 and 3.1. The chip designer's suggestion will be used as a reference in his thesis to generate this pulse. Using the current equation 3.1 will consist of use one capacitor to generate the charge.

On equation 3.1, we can see that the current in the (ideal) capacitor depends on two factors: C (capacitance of the capacitor) and $\frac{dV}{dt}$. To regulate the current pulse, we must control at least one of these factors.

4.2.1 Different voltage values for charging C

Using electronic keys, it is possible to charge and discharge a capacitor in a controlled way. The capacitor, under certain conditions, must be connected to a voltage V, charged, and therefore generate a current $i = C\frac{dV}{dt}$ (accumulating a charge $Q = CV$, and then discharged to the ground.

To achieve this charge and discharge, two digital signals will be used to control a pair of electronic switches (Mosfets). These transistors will have the function of controlling the charge and discharge of the capacitor.

Figure 4.4(a) shows the circuit at the transistor level where $M1$ and $M2$ are the electronic keys, clk_1 and clk_2 are the control signals, V is the voltage source, C_x is a capacitor in charge of accumulating charge that will be delivered to the chip (R_{chip}), R_{dac} is the equivalent resistance of the internal DAC of the chip channel, C_a is the coupling capacitor mentioned in section 2.3 (Fig 2.4) which must have a value of $1uF$ and R_{Chip} is the equivalent input resistance ASIC (11 ohm) Altamirano (2021)

Figure 4.4(b) shows the control signals. The figure 4.4(b) and 4.4(d) shows the circuit of the Fig 4.4(a) at the switch level. Where the first stage (b) is the charge and delivery of the current pulse to R_{chip} and stage (d) is the discharge of capacitors to the ground.

Note that C_x and R_{dac} make up a high pass filter, and C_a and R_{chip} are another high pass filter (see Fig 4.4 (a,b,d)). Where the equivalent transfer function for these two filters in series will be:

$$Hs_1 \cdot Hs_2 = \frac{s}{s + \frac{1}{R_{dac}C_x}} \cdot \frac{s}{s + \frac{1}{R_{chip}C_a}} \tag{4.6}$$

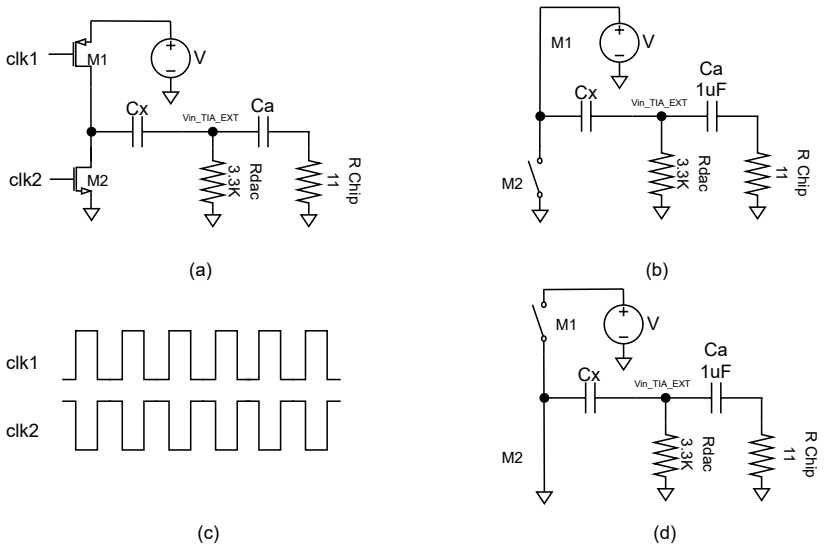


Figure 4.4: (a) Circuit at the transistor level-view. (b) 1st stage, charging the capacitors. (c) Digital control signals(referential). (d) Second stage, discharging capacitors to the ground.

And for example, supposing that we assign a value of $350pF$ to C_x, the transfer function is as:

$$Hs_1 = \frac{s}{s + 8.658 \cdot 10^5} \tag{4.7}$$

$$Hs_2 = \frac{s}{s + 9.091 \cdot 10^4} \tag{4.8}$$

$$Hs_1 \cdot Hs_2 = \frac{s^2}{s^2 + 9.567 \cdot 10^5 + 7.871 \cdot 10^{10}} \tag{4.9}$$

Analyzing the resulting system, we can see that the function poles are distant, not interfering with each other. We can also notice that the pole of the filter generated by C_a and R_{chip} will not affect the current pulse(Fig 4.5). The latter is reinforced when analyzing the frequency behavior of both

filters in series (Fig 4.6), considering that the current pulse will be above the frequencies of $10^8 Hz$.

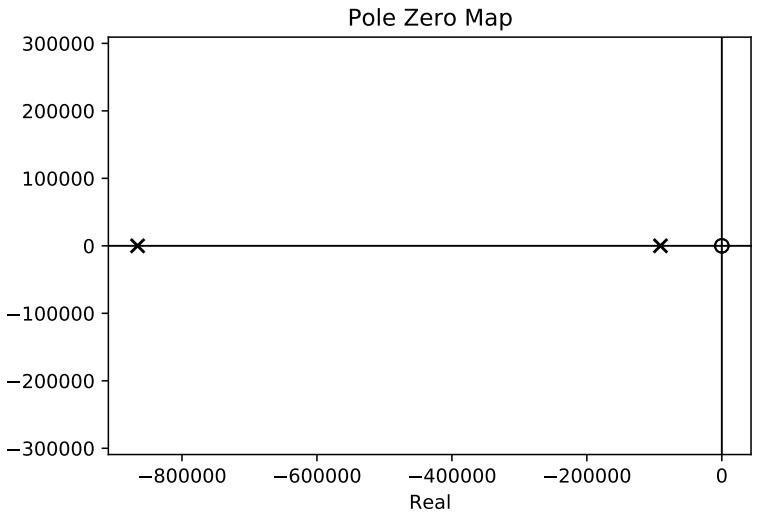


Figure 4.5: Map of poles and zeros of the equation 4.9

The capacitor C_x must accumulate a charge Q according to the equation of current $Q = CV$, where the extremes of the values (minimum and maximum) are found in Table 3.1. Considering the previous equation, we note that a voltage V must be selected (which corresponds to the voltage V of Fig 4.4a). This voltage must be set to different values so that from an exact value of C_x, different values of charge Q can be generated. They are many forms to set a voltage, it can be used from a voltage divider adjusted with a potentiometer to voltages set by a DAC (Digital to Analog Converter) to establish this voltage. The DAC option will be chosen because it has precision, output stability and monotony assured by the manufacturer, fundamental parameters if we are looking for accuracy in the generation of Q charge.

For the election of the DAC, we must ensure some specific parameters.

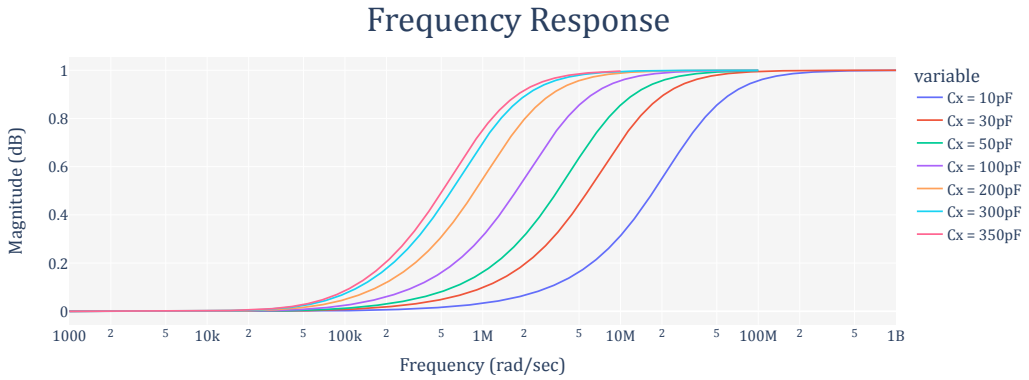


Figure 4.6: Frequency response pulse current circuit generator for different values of C_x

The resolution, the maximum voltage delivered, maximum short-circuit current are examples of some parameters.

From an application view, the resolution of the DAC depends on the number of bits and of the reference voltage. For example, considering a 4-bit DAC (Fig 4.7, Table 4.1) and considering a reference voltage $V_{ref} = V_{ref} = 3.3V$. The resolution will be given by the equation 4.10.

$$Resolution = \frac{V_{ref}}{2^n - 1} \tag{4.10}$$

, resulting in a minimum step of 0.22V. Another example and using the same equation 4.10 but in this case for a 12-bit DAC, the minimum step-size (resolution), with $V_{ref} = 3.3V$, will be $0.8mV$.

The converter's resolution will be essential so that using the charge equation $Q = CV$ it is possible to generate voltages that allow having load values equal to or less than one pixel (4E-13, Table 3.1). Assuming a 12-bit DAC

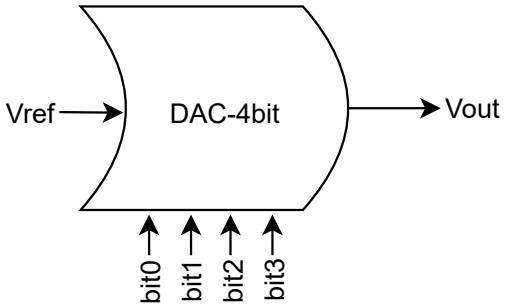


Figure 4.7: Example of 4bit-DAC

Table 4.1: 4bit-DAC reference outputs

Bits	**Vout**
0000	0
0001	0.22
0010	0.44
0011	0.66
...	...
1101	2.86
1110	3.08
1111	Vref

$C_x = 300pF$, $V_{ref} = 3.3V$, the minimum charge generated will be:

$$Q_{min} = V_{min} \cdot C_x = 0.8mV \cdot 350pF = 2.4E - 13 \tag{4.11}$$

. Resulting in a charge value of less than one pixel 3.1.

The highest voltage delivered by the DAC will be approximately equal to the reference voltage of the DAC. It is relevant to consider this maximum voltage since if one seeks to comply with the maximum charge generation expressed in Table 3.1, C_x is fixed. This voltage must be greater than or

equal to:

$$V = \frac{Q_{max}}{C}, C = C_x \tag{4.12}$$

For a maximum charge value of 1.2E-9 and $C_x = 350pF$, it is $V = 3.4V$. Therefore, for a fixed C_x value and to generate the maximum load from Table 3.1, a DAC must deliver a maximum voltage greater than or equal to $3.4V$.

The short-circuit current is a value given by the converter manufacturer in the datasheet. This parameter is vital in order not to damage the DAC when the current pulses are generated. Based on Spice simulations, a short-circuit current value greater than 200mA is recommended.

The circuit of Fig 4.4(a) was simulated in Lt-Spice with commercial transistors, considering various input voltages from $100mV$ to $3V$with a step of $200mV$. Control signals were shown in Fig 4.8. In Fig 4.9 the voltage corresponding to the connection node between the source and the source of the transistors M1 and M2, respectively, is shown. It is important to note that these transistors are operating in Triode-region because $V_{ds} \leq V_{gs} - V_t$.

Fig 4.10 shows the current pulses generated by the clk1 and clk2 control signals. Note that there are negative and positive currents. In the simulation, the current from *Rchip* to the circuit was measured; therefore the negative currents go to the chip.

4.3 Control and reading mechanisms for output data.

The chip outputs are fast, on the order of nanoseconds. To implement the estimation methods presented in Chapter 3, one must be able to correctly

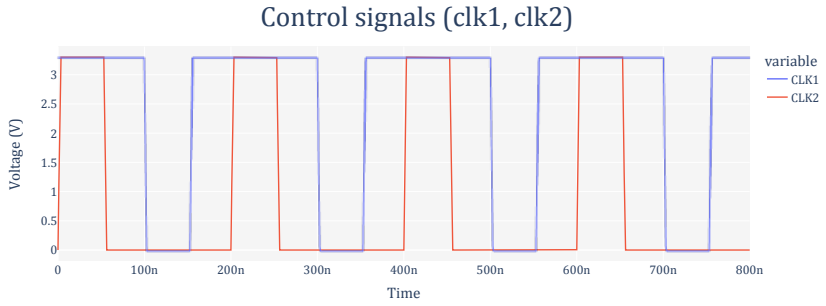


Figure 4.8: Spice simulation, control signals for M1 and M2

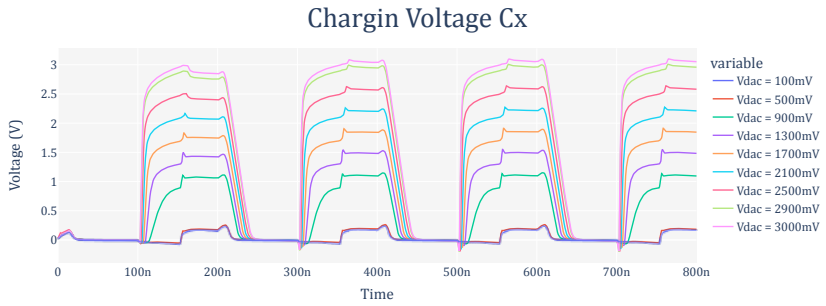


Figure 4.9: Spice simulation, charging voltage (common node between transistors M1 and M2)

read and process the information provided by the circuit outputs.

Analyzing post-layout simulations, the following questions arise:

1. How to read the signals of Fig 3.1, 3.2, 3.3

2. How do you know when to read?

3. How much information is necessary to capture to proceed with the estimation methods mentioned in section 3.2?

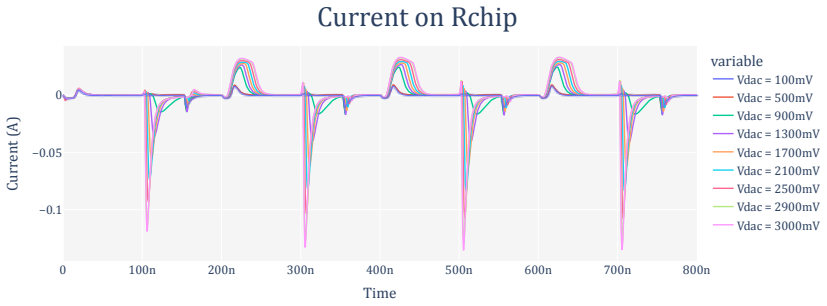


Figure 4.10: Spice simulation, current to the chip

4.3.1 How to read the output signals of the chip?

To answer question this question, it is necessary to remember that the circuit has digital ($Vout\_disc$) and analog outputs ($Vout\_TIA$, $Vout\_INT$). The mechanism for read this signals will be different and it'll be described in this section.

Digital Outs

The digital outputs deliver information by changing their logic state (on/off transition and vice-versa). The information provided by this signal is the time of occurrence of the arrival of photons to the detector and also can be used for extract information about the approximate number of activated pixels. To detect the time of occurrence (pulse event), it is necessary to identify where the digital signal changes from high to low. Correctly capturing this information requires an additional circuit that detects with sufficient temporal definition ($< 5ns$ according to simulation data shown in Table 3.2) the on/off transition moment. To estimate the number of activated pixels, it is necessary to count how long (after the on / off transition) the discriminator signal is in a low state before changing to a high state. The duration of

the $Vout\_DISC$ signals will depend proportionally on the amount of charge read by the chip. In Table 3.2, you can read the results for specific numbers of activated pixels. Therefore, correctly capturing this information requires an additional circuit that detects this event with sufficient resolution.

Analog

The signals *Vout_ TIA* and *Vout_ DISC* have information encoded in amplitude, that is to say, that the maximum value measured from its initial value provides information. The data delivered by these signals correspond to the number of activated pixels and was explained in more detail in chapter 3. These analog signals are non-periodic events, and their moment of occurrence can be controlled by the activation of the charge generator circuit. Like the *Vout_ DISC* outputs, the duration of the *Vout_ TIA* and *Vout_ DISC* signals will depend proportionally on the amount of charge read by the chip. The results recorded in the post-layout simulations shown in Fig 3.1, 3.2, provide us with relevant information regarding the scenario of a triggered pixel. This event will produce the fastest and lowest amplitude outputs.

4.3.2 How do you know when to read?

To answer the second question, which mentions knowing when to read, it is necessary to remember that the charge generation process can be controlled in its entirety, so it can be predicted when there will be relevant information to read from the $Vout\_DISC$ and $Vout\_TIA$ outputs. This means that it is not necessary to read or monitor the analog outputs permanently. It must also have a controllable reading system, that allows to start and end the

reading at certain times. For the digital signal of the discriminator circuit, it is only necessary to have a circuit that detects the transition moments *on-off* and *off-on*. See Fig 4.13.

4.3.3 How much information is necessary to capture to proceed with the estimation methods mentioned in section 3.2?

Question three refers to how much information is necessary to use the methodologies outlined in section 3.2. In Altamirano (2021) it is recommended to use the second approach (max amplitude) to take full advantage of the capabilities of the ASIC. For this reason, we will focus on this methodology and how it is possible to obtain enough information to implement it.

The approach called **Maximum amplitude** requires a reading system capable of recording the maximum amplitude of analog signals. To achieve this, it is necessary to digitally store the analog information, allowing its later analysis. This analog-digital conversion necessarily requires a specialized circuit called analog to digital converter.

About the Analog to Digital Converters

Analog to Digital Converters (ADC) are circuits that belong to mixed-signal circuits, and these devices transform the analog signal into a digital signal using different techniques. There are various types of ADC, each architecture employing different methodologies to achieve the conversion. Some of these ADC classes are Flash, SAR, Delta-Sigma. Regardless of the type of ADC, their functionality is to transform a continuous analog signal to a discrete

signal composed of digital samples (Fig. 4.12). Every ADC performs at least three processes. These processes are:

- Analog signal sampling.
- Signal quantization.
- Signal encoding.

The sampling process depends on the sampling period of the ADC (generally expressed in samples per second, SPS). This parameter is relevant when selecting an ADC. Depending on the maximum frequencies of the analog signals, at least twice the maximum frequency (Nyquist) must be sampled to recover a signal and avoid Aliasing. In practice, it is recommended at least ten times the Nyquist frequency.

The quantization of analog signals is the assignment of discrete values to the samples acquired in the sampling process. This quantization depends directly on the number of bits that the ADC has. For example, an 8-bit ADC has 2^8 possible values to assign to the sampled values. These 2^8 possible values depend on a reference voltage, this voltage being an input of the ADC. In general, the quantization process depends on the ADC selected, and the manufacturers explain these details in the device datasheets.

An example of 4bit ADC is shown in Fig. 4.11

The quantization process involves a quantization error. This error has a form of random noise. It is uniformly distributed around $\pm\frac{1}{2}LSB$ with zero mean and standard deviation given by the equation (Lutenberg (2012)):

$$\sigma = \frac{1}{\sqrt{Nbits \cdot LSB}} \tag{4.13}$$

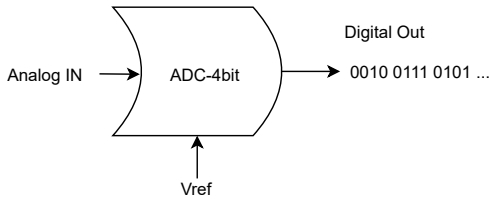


Figure 4.11: 4bit ADC. High level view.

LSB is the Least Significant Bit (quantization step), which corresponds to the distance between two adjacent levels in a quantization. This error implies in certain situations an approximation when quantizing. For example, when the sample is between two values (of the 2 Nbits), it should be approximated to the closest (See Fig. 4.12). This error decreases according to equation 4.13 as the number of bits in the converter increases. The encoding consists of transforming the quantized values to binary digital values, and that can be stored in a memory or be delivered to another digital circuit.

In addition to these parameters already mentioned there are others, such as the maximum bandwidth of the converter, a parameter that indicates up to which frequencies the attenuation will be less than $-3dB$. We also find the acquisition time, a parameter expressed in units of time and refers to how many units of time after sampling the signal are quantified.

In our case, the analog signals are of the order of nanoseconds, which, based on *Nyquist*, would imply selecting an ADC of the order of GSPS, in addition to having sufficient bit resolution to quantify the samples accurately and have a bandwidth higher than the $200MHz$, so that the Discriminator and Integrator signals are not attenuated. Using an ADC with the characteristics mentioned above would work to obtain enough information to allow the application of the maximum breadth methodology, but the associated cost is

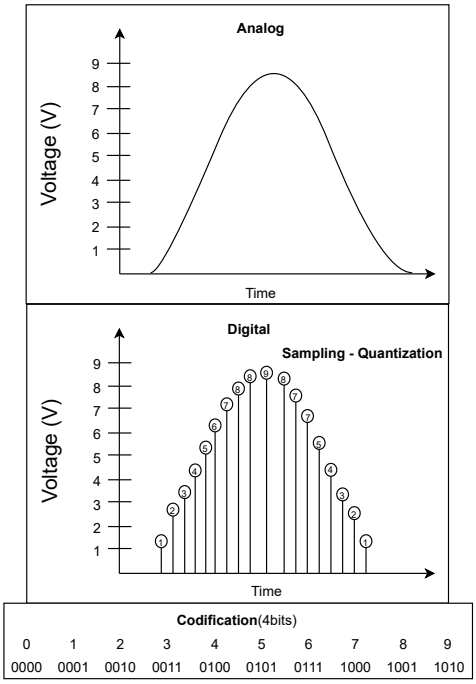


Figure 4.12: Analog to digital conversion example.

high. Remember that the generation of analog signals depends on the charge generator circuit, which makes the process controllable and reproducible in time. Considering this, a technique called sub-sampling can be applied. This technique will be described in section 4.3.4.

4.3.4 Sub-sampling

This technique consists of sampling a signal (which is fast for the acquisition system) repeatedly at different points in time. This technique allows the reconstruction of the original signal without the need of fast ADC. Applying this technique depends on knowing, and being able to control, the moment of arrival of the signals to be processed. They must also have precise control and reading mechanisms. The ADC to implement this technique can be

slower than that required by the Nyquist theorem, but it must guarantee a fast acquisition time that allows sampling at precise moments and a correct bandwidth so as not to attenuate the chip signals.

For the characterization of the ASIC, the charge generation process is controllable and repeatable, so it is possible to implement this technique with the appropriate elements.

In Fig 4.13 three graphs are shown in time, where the upper graph is the trigger signal for the charge generator circuit, the middle signal is a representation of one of the chip outputs, and the graph below It is the response of the digital output of the ASIC when the readed charge by the chip is equal or greater than the equivalent charge of one activated pixel.

The sub-sampling process of the signal in the middle graph is graphically expressed in Fig 4.14, wherein Fig 4.14(a) is shown at a certain point in time, Fig 4.14(b) at another, and so on until the different samples can reconstruct a significant part of the original signal (Fig 4.14(f)). Each of the samples can be an average of samples taken simultaneously to correct any noise that may exist.

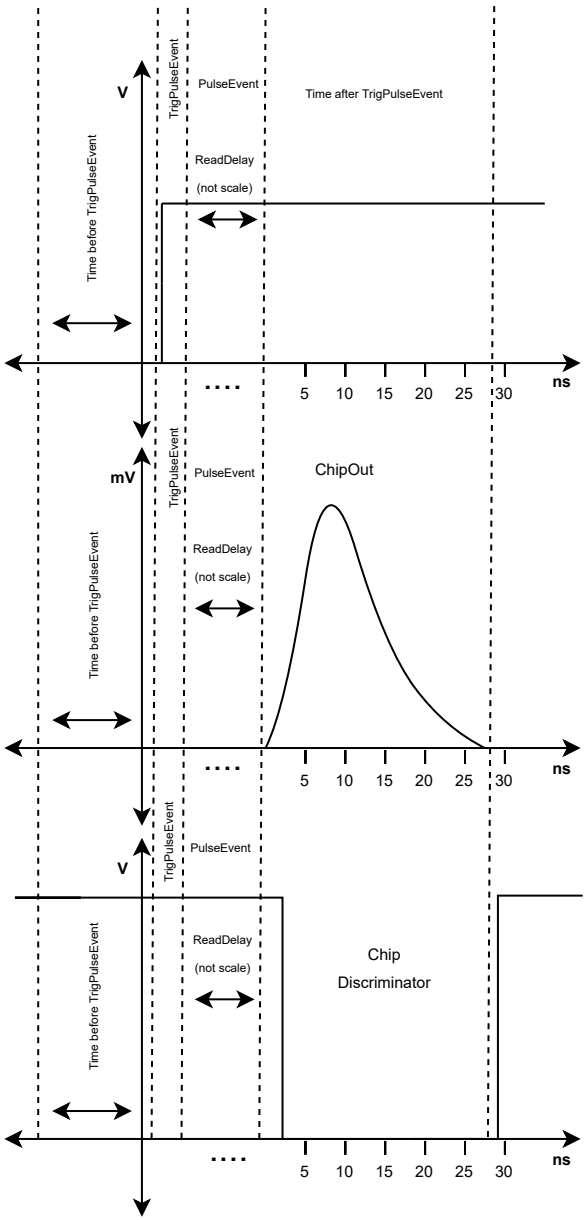


Figure 4.13: Succession of events in time

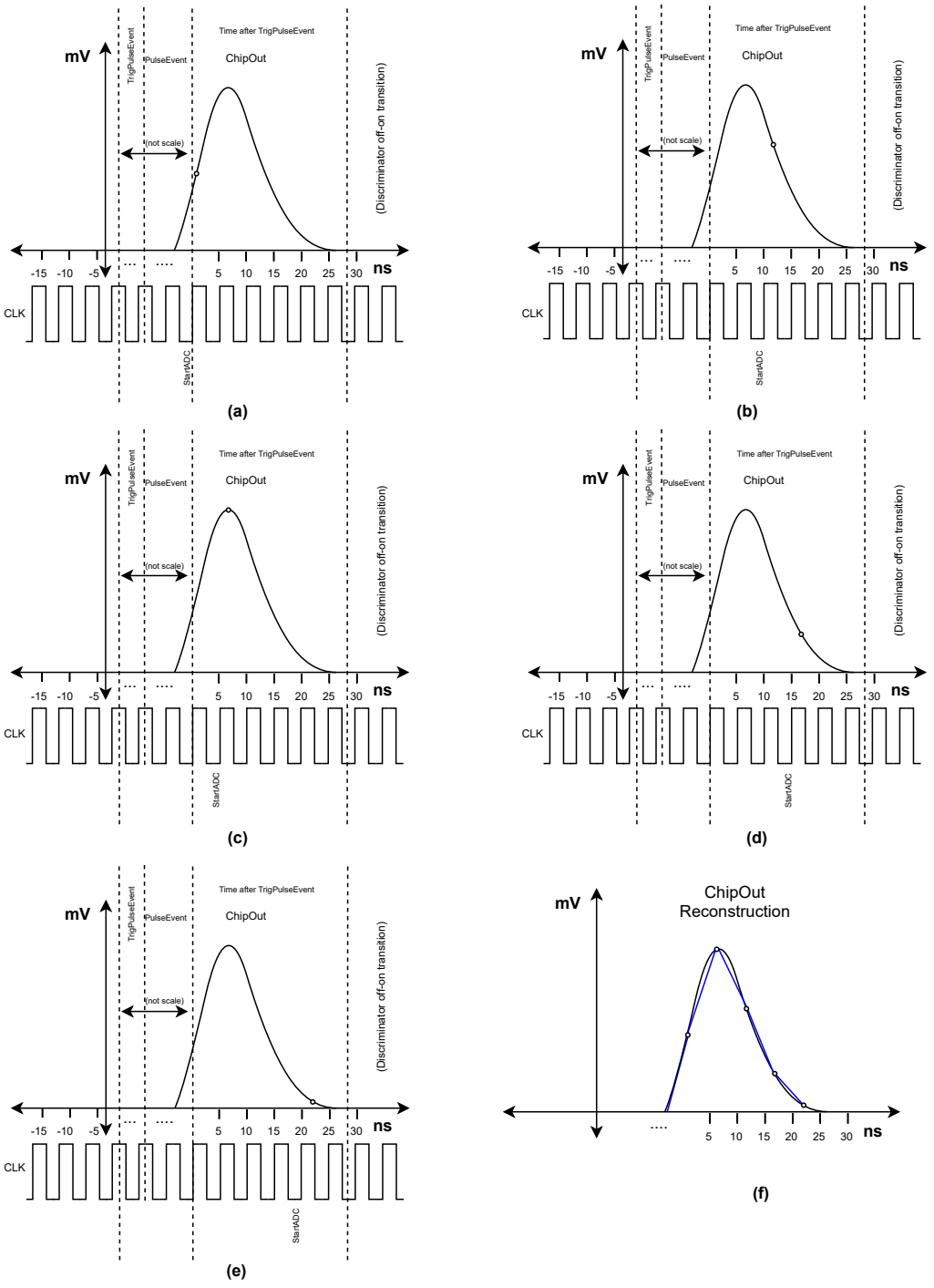


Figure 4.14: Sub-sampling methodology.

5 Design and documentation of the test environment

To implement the methods described in the previous chapters, it is necessary to have a system that allows the control of events with nanosecond precision that provides flexibility and re-configurations in some instances. Given the requirements, the use of an FPGA (field-programmable gate array) is proposed to implement the control mechanisms of the circuit characterization and verification process. The FPGA chosen is part of the Zynq-7000 family (Zybo Board).

In addition to the FPGA, it is necessary to have a printed circuit board designed to contain all the components that will be necessary to read and stimulate the chip. This PCB must allow communication between the FPGA and the components of the printed circuit and vice versa.

5.1 General description of the test environment

A global view of the test system can be seen in Fig 5.1; it is possible to see the MPPC detectors, the Chip and each of its channels, the outputs, a signal conditioning block, and finally, the FPGA. In the initial verification process, the MPPCs will be replaced by the charge generator circuit of section 4.2.

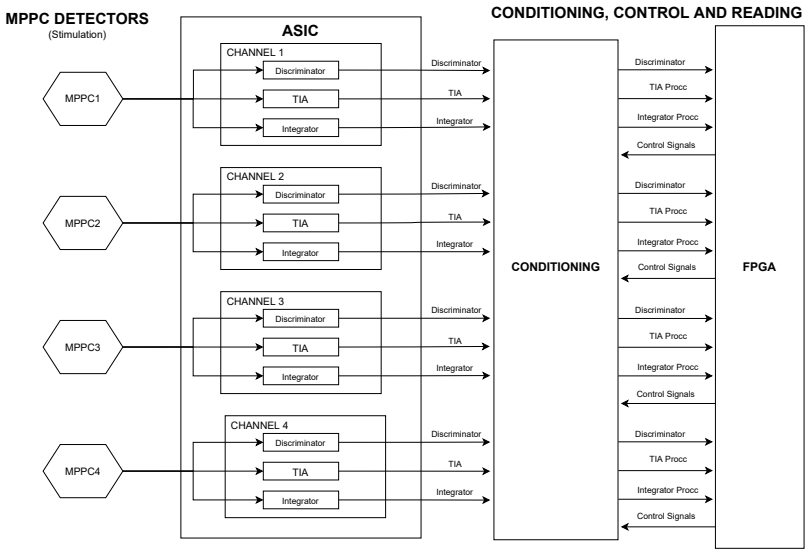


Figure 5.1: Global view of the testing system.

This circuit will be part of a PCB where other elements necessary for the test process will be soldered, the elaboration of this board and its specific components will be soldered. Will be presented in a later chapter.

The conditioning stage includes the analog to digital conversion of the analog signals of the Chip as well as the DAC that allows controlling the voltage V of the charge generator circuits described in section 4.2, Fig 4.4.

It is also essential to consider that the chip has 12 analog outputs, and each one delivers information. According to the requirements of this thesis, it is not necessary to read all these signals in parallel, which allows only one ADC to be used for the entire system. However, a suitable multiplexer is required to achieve those mentioned above. The multiplexer must have a low input capacitance and a bandwidth large enough not to attenuate the chip-out signals. Reading the digital signals (discriminator circuit out) requires a digital circuit that will be implemented in the FPGA, which will detect

the transition from high to low state and count how many times elapses will elapse until the transition from low to high state. The definition of the account of this circuit will be proportional to the FPGA clock speed. In this case, it will be 5ns, justified time if the data in Table 3.2 are analyzed.

In addition to the counter circuit, the FPGA must have other custom logic blocks, which allow, for example, to store information from the ADC quickly (on each clock edge) and be delivered to the user when requested. There must also be logical blocks in charge of controlling the converters (ADC-DAC) to control their operation according to the mechanisms described in the datasheets of each component. Remember that the ASIC also has 4 DACs, one for each channel. The control of these blocks must also be considered in the custom blocks to be designed. In Fig 5.2, the conditioning stage and internal blocks of the FPGA can be seen in more detail. Note that the FPGA has two internal blocks, Custom logic, and Zynq Processor System. Zynq Processor System is the Xilinx manufacturer's designation for its SoC (System on Chip). In the words of the manufacturer: "Xilinx provides the Processing System IP Wrapper for the Zynq®-7000 to accelerate your design and its configuration for your embedded products. The Processing System IP is the software interface around the Zynq-7000 Processing System. The Zynq-7000 family consists of a system-on-chip (SoC) style integrated processing system (PS) and a Programmable Logic (PL) unit, providing an extensible and flexible SoC solution on a single die. The Processing System IP Wrapper acts as a logic connection between the PS and the PL while assisting you to integrate custom and embedded IPs with the processing system using the Vivado IP integrator."

Using this system, it is possible to interact with different peripherals of the ARM Cortex-A9 processor(UART, SPI, GPIOs) to facilitate the acquisition and control of custom logic through the AXI bus.

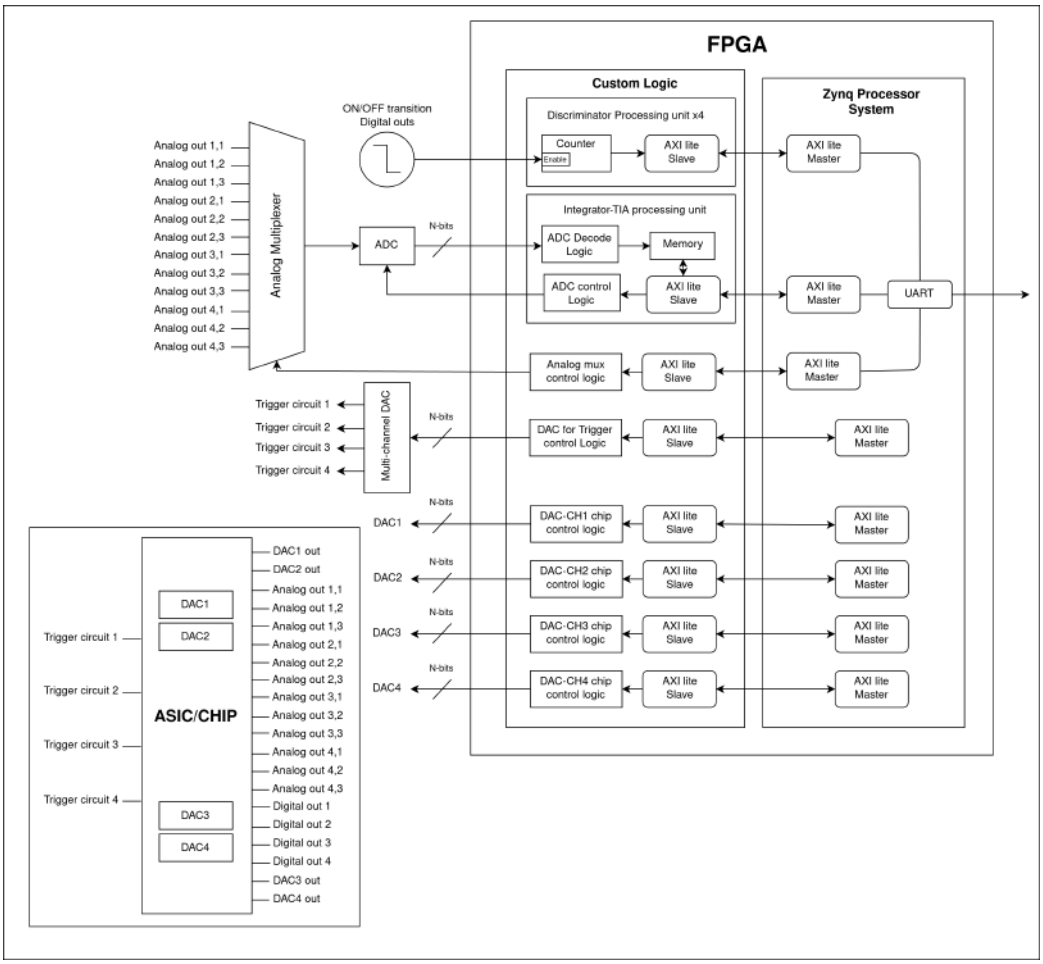


Figure 5.2: System implementation overview.

5.2 PCB design and implementation

The PCB was designed to contain all the electronic elements and components necessary to have a robust testing environment for the chip, these elements

and capabilities were described in the previous chapter.

The design process of any printed card begins with the specifications, to continue with the steps of: schematic creation, selection and assignment of physical components, generation of the netlist, placement of the elements, routing, verification and finally manufacturing. To comply with the previous stages, programs of the CAD (computer aided design) category are used, in this particular case the KiCad software was used.

Of the steps mentioned above, the specifications have been mentioned in the previous chapters and these specifications are decisive in the selection of the components. Last but not least for this selection step it is important to consider the availability on the market of these elements.

5.2.1 Components

If we divide the PCB elements into groups based on their purpose, we can mention the following:

- **Stimulation components:** Charge generator circuits (CMOS switches and capacitors), DAC to adjust the voltage of the capacitors used in charge generation, and FPGA control inputs for control purposes of the elements of this group.

- **Reading components:** Digital analog converter for reading the analog outputs of the chip, operational amplifiers in charge of processing the differential outputs of the integrator circuits, and inputs/outputs (IO ports) of the FPGA for the control of the ADC and for reading of digital outputs (discriminator circuit) ASIC.

- **Control components:** Analog multiplexers for the selection of outputs/inputs of the chip to be read and/or stimulated and control inputs of the FPGA for control purposes of the elements of this group.

- **Power components:** Power supplies of various values to power elements of the previous groups. It is important to mention that there are two types of sources: digital (FPGA) and analog (PCB modules).

5.2.2 Placement and routing

The placement stage is the physical/spatial location of the PCB components. The components must be distributed based on different criteria such as, for example: the specific function of these elements and the relationship between them, analog or digital domain, noise sources and circuits that are high sensitivity to the noise (power sources and converters).

The routing stage considers the physical interconnection of the elements on the printed circuit, these being power supply interconnections and input or output signals, which can be analog or digital. For the development of these stages, the effects of having elements with digital and analog signals coexisting in the same circuit (considering the PCB as a mixed-signal circuit in itself) at the same time must be carefully considered. There are technical guides for the application and design of mixed signal PCBs, in which details and advice are mentioned when designing this type of electronic card Pithadia and More (2013). The detail that stands out the most in these documents is related to the ground of the circuit and if it is necessary to have a ground for the digital signals and another for the analog ones, in order to prevent the current returns of the digital signals from interfering with the analog ones.

thus introducing distortions in the measurements. In this section there are three possibilities:

- Two different grounds, one analog and one digital (Figure 5.3)
- Two different grounds, one analog and one digital but connected at a single point on the PCB (Figure 5.4)
- One solid ground plane, without jumps or gaps (Figure 5.5)

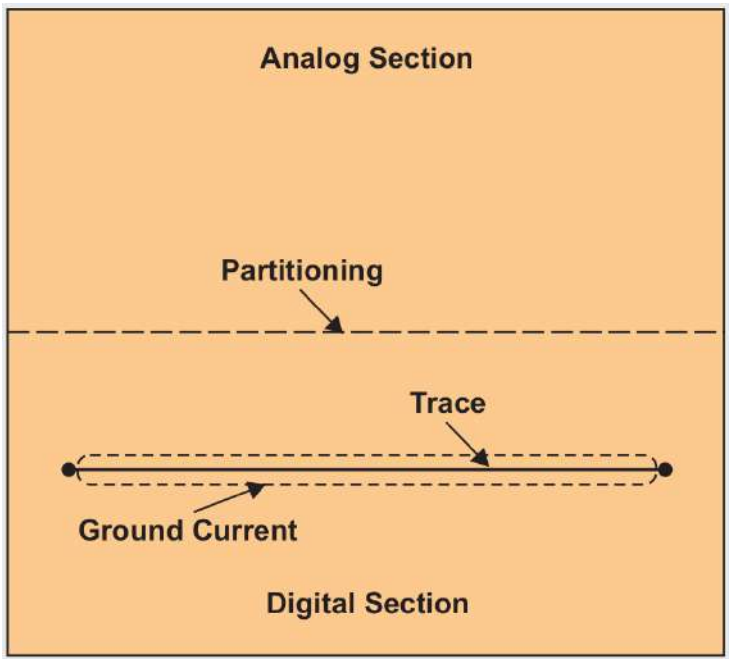


Figure 5.3: PCB with one solid ground plane

The option that, according to the application articles, gives the best results is the option of only one ground plane as long as the placement and routing of the components is correct. If we opt for the option of one ground plane, at the placement level, a virtual separation of the components must be

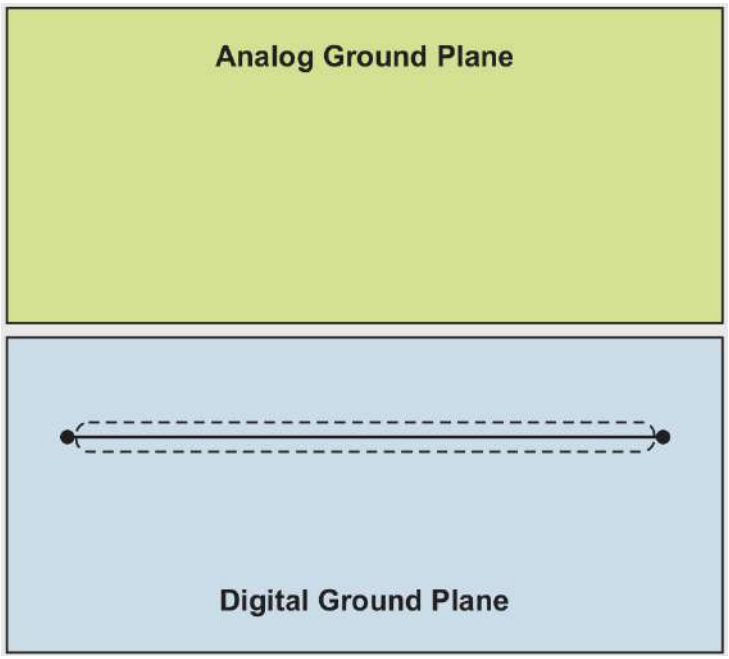


Figure 5.4: PCB with two ground planes (analog and digital planes)

made, maintaining a common ground for both digital and analog elements. This virtual separation consists of spatially separating the purely analog elements from the digital ones and using the PCB subcircuits that are mixed signals (DAC's, ADC's) as reference elements for the virtual separation. In the routing stage, extreme care must be taken to respect the virtual separation made in the placement, this means that digital signals, especially fast ones, for example communication tracks or clocks, must not cross the analog zones. Regarding the power supplies, an exclusive layer is recommended for the power supplies. In this case, the PCB has a dedicated power supply layer, with a digital and an analog power supply section.

The placement and routing of the components can be seen in figure **??** and in figure 5.7 the partition of the components into analog (red) and digital

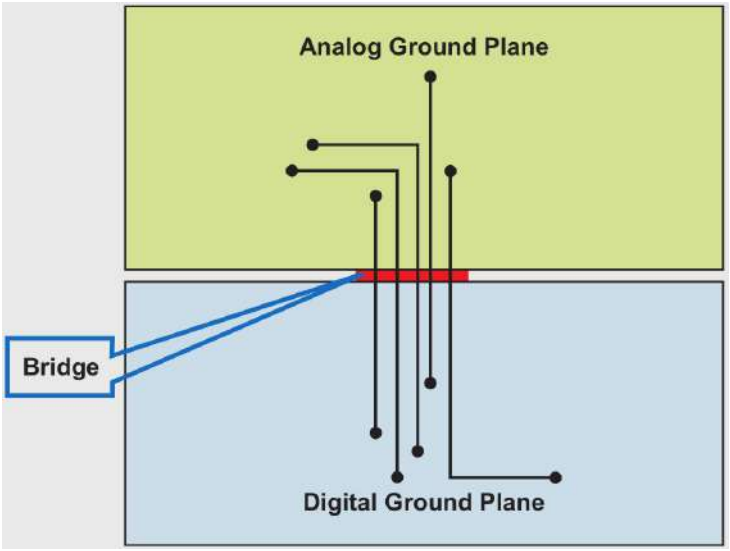


Figure 5.5: PCB with two different grounds connected at a single point

(blue) sections can be seen. Figure 5.8 is a 3D rendering of the PCB once the routing and placement stages have been completed.

5.2.3 Verification of design and manufacturing rules

The manufacture of the PCB is carried out by a specialized technology company. To achieve correct manufacturing, the design rules and manufacturing capabilities of the factory must be complied with. Depending on the program used in the design, these rules can be established in the software and verified before being sent to the company in charge of manufacturing the PCB.

The factory will only carry out the manufacturing process if the design rules imposed by them are satisfied (manufacturing constraints).

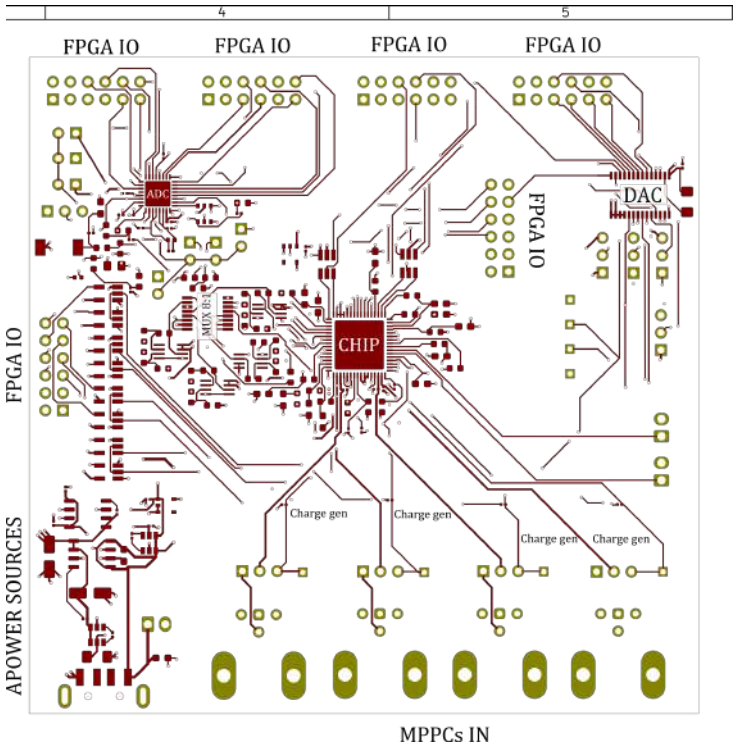


Figure 5.6: Placement and routing of the PCB components

5.3 FPGA Programming: Design and RTL simulation of the digital modules

The FPGA has the mission of controlling the PCB and their different components. For this, it must have the necessary control, reading and stimulation logic and every of this logic needs to be designed, programed and implemented on the FPGA. These steps will be divided into two principal stages:

1. Design and verification of digital modules (IPs) dedicated to control, reading and stimulation logic.

2. Incorporation and integration of IPs with AXIbus and Zynq processor system.

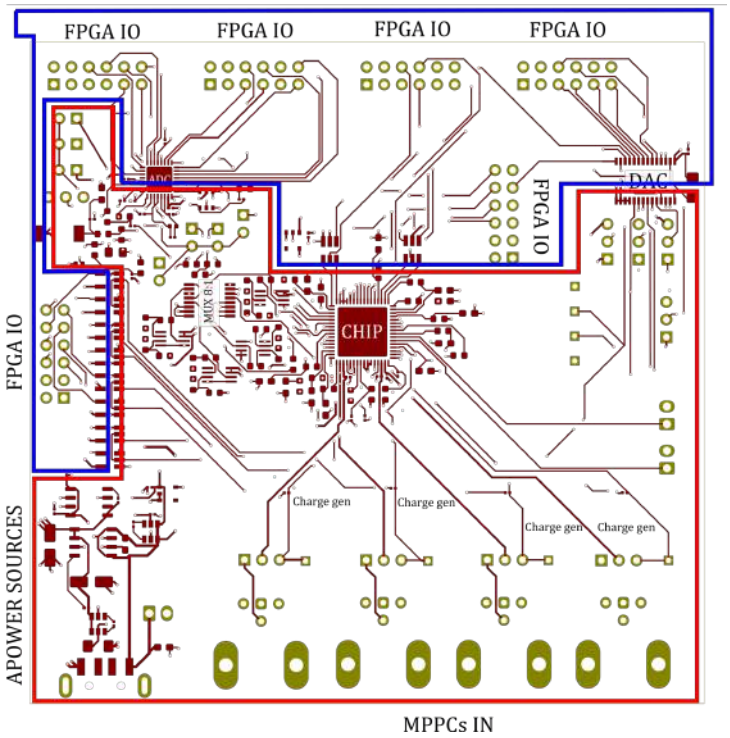


Figure 5.7: Analog and digital zones of the PCB

This stage involves the design and verification of digital circuits in some HDL (hardware description language). The purpose of these circuits is varied, so we will divide them into four blocks:

- ADC module and trigger system
- DAC module
- MUX module
- Discriminator counter module
- Chip state machine module

Each of these blocks is an independent digital circuit.

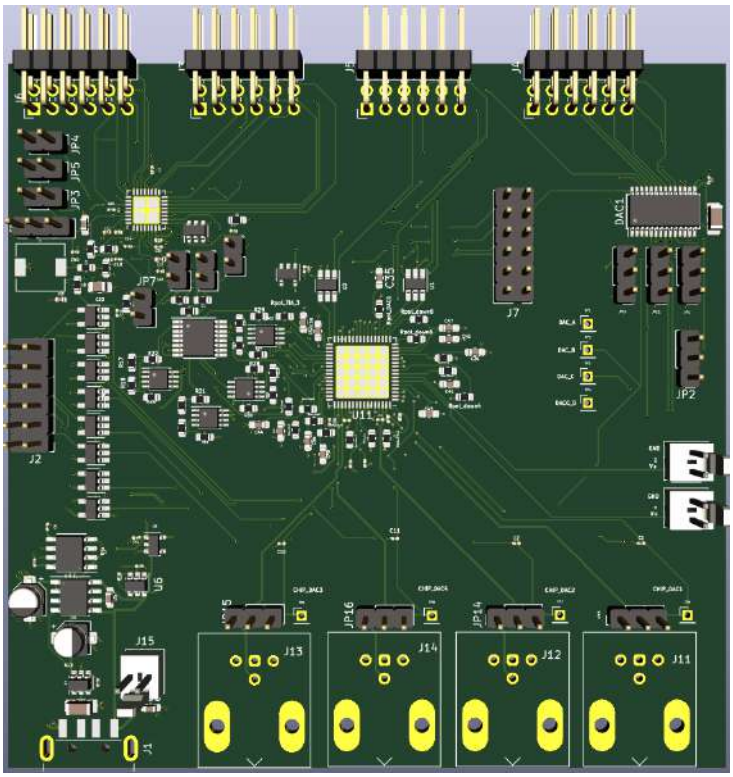


Figure 5.8: 3D rendering of the PCB

5.3.1 ADC module and trigger system

This digital module contains all the logic necessary to configure some aspects of the ADC, it also contains the trigger logic of the charge generators, as well as everything necessary to implement the subsampling technique mentioned in the previous chapter.

On the one hand, the circuit is made up of a section dedicated to dividing the FPGA input clock frequency (200MHz) and reducing it to 10MHz, the latter being the maximum operating frequency of the ADC.

On the other hand, a series of internal counters were designed that serve to precisely control the triggering moments of the load generating circuits as

well as when to store the ADC samples. This synchronization is essential when implementing the subsampling technique since at all times we must know when to sample and when to alter the state of the circuits that generate the load. This circuit can work in two ways: single shot or multi shot.

Considering the multi-shot configuration, figure 5.9 shows graphically the way of implementing the subsampling technique. Starting with an ADC configuration setting time defined in 5 clock cycles, then a time window defined as "trigger window" is defined, which is of a maximum duration of 100ns where the trigger of any of the charging generators. Now if the trigger is performed at 600ns (considering the 500ns setting and 100ns trigger window) we can consider that the sample taken at 600ns will be our s1 (figure 5.9b), then we can repeat the setting process and trigger 5ns before the end of the trigger window, thus resulting in sample s2. The process can be repeated in a similar way for s3, s4 and s5. Figures figure 5.9b and figure 5.9c show this behavior graphically. It is important to mention that each sample s1, s2, s3, s4 and s5 can be taken N times and averaged at the end of the sampling cycle (when the last sample s5 is captured).

This IP is also capable of selecting which trigger circuit to use, since there are four independent circuits. The charge and discharge states of the capacitors are controlled in such a way that they do not affect the measurements of the ADC while it acquires the value and does the conversion.

The figure 5.10 demonstrates at the simulation level the sampling of the first sample *s1* (this being the value 4)

The figure 5.11 demonstrates at the simulation level the sampling of the first sample *s2* (this being the value 4), let us note that here the trigger firing

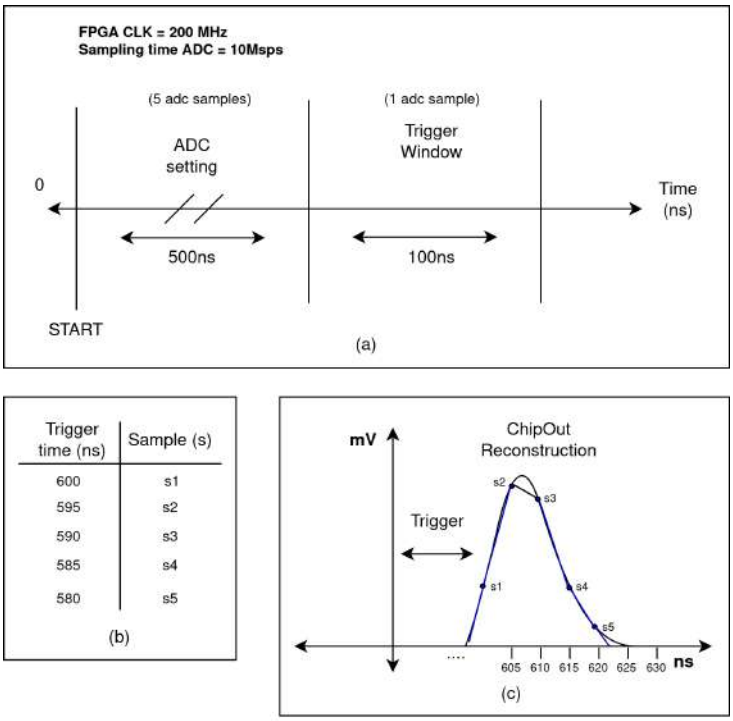


Figure 5.9: Graphical description: how implement the subsampling technique?

is 5ns earlier than in the case of *s1*

The figure 5.12 demonstrates at the simulation level the sampling of the first sample *s3* (this being the value 4), let us note that here the trigger firing is 10ns earlier than in the case of *s1*. Similarly, this occurs for the rest of the samples *s4* and *s5*.

In the figure 5.13 we can observe the successive and repetitive sampling (in this case simulated with N=4) for the sample s1 to s4. In the figure 5.14 we can observe the sampling for *s5* and the activation of the *end_flag* signal, at this point the average of the samples *s*1, *s2*, *s3*, *s4* and *s5* are calculated.

Now if we consider the single shot configuration, in this case only one shot will be made at 600ns. This operating mode will be useful when looking

Figure 5.10: ADC module simulation result: sample s1

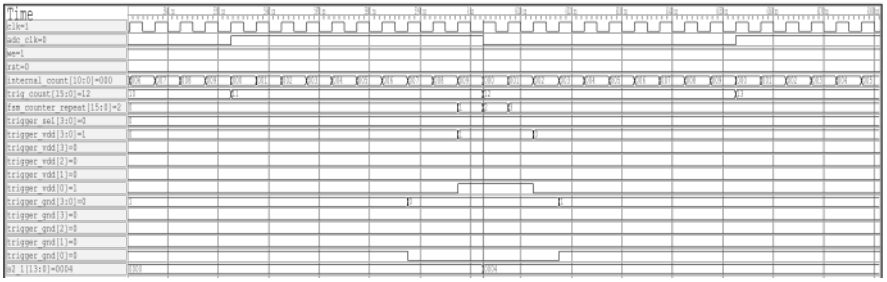


Figure 5.11: ADC module simulation result: sample s2

to measure with the discriminator circuits. This module is shown as a block graphically in figure 5.15 and in this figure is possible to see the inputs and outputs of the ADC module.

5.3.2 DAC module

This module is designed to set a voltage value on any of the DAC outputs. Internally, a conversion is made from a serial value (input, desired voltage level) to an output value that will represent bits(serial to parallel converter). A diagram of this module can be seen in figure 5.16 . A RTL(Register Transfer Level) simulation of this module was carried out, and its behavior is shown in 5.17 where a serie of serial-inputs values was used like inputs, obtaining the binary value corresponding to the entered serial value.

Figure 5.12: ADC module simulation result: sample s3

5.3.3 Counter module for discriminator circuits

The chip has 4 outputs of the discriminator circuits, these outputs when detecting the arrival of one or more photons will change their state from high to low. The time that these outputs are in a low state is directly proportional to the number of photons that arrived at the detector Altamirano (2021). This module performs the function of counting how many clock cycles the discriminator outputs are in a low state, the accuracy of this counter is $5ns$, limited by the FPGA clock frequency. The module has 4 independent counters that can work in parallel, which will always have a transition from high to low state in their respective inputs. A diagram of this module can be seen in 5.18. A RTL simulation of this module was carried out, and his behavior as shown in figure 5.19 where the operation of two counters is observed when their inputs change respectively from a high state to a low state. The counters stop running when the respective input signal returns to its high state.

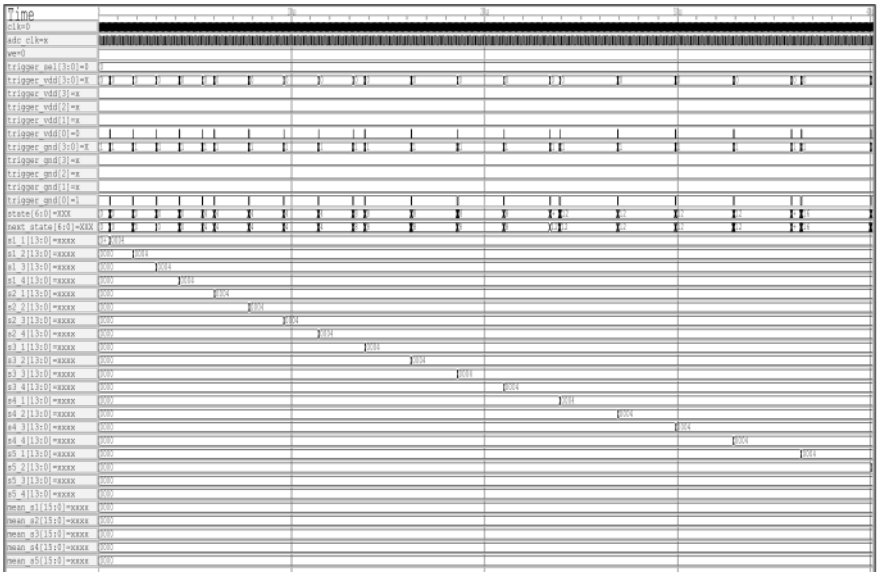


Figure 5.13: Successive and repetitive sampling for the sample s1 to s4

5.3.4 Chip state machine

The chip has a state machine dedicated to the configuration of its internal DAC's. This state machine allows independent voltages to be established for each of the 4 DAC's, thus allowing the detectors to be accurately biased. A diagram of this module can be seen in 5.20.

To configure this state machine, a digital logic must be designed in the FPGA that allows data to be entered into the state machine of the chip. This digital logic was designed based on the specifications given in Altamirano (2021), and consists of a clock generator module with a period of $20uS$, a state machine that allows configuring a 3-bit address corresponding to the address of the DAC to be written and which also allows the selected DAC to be set to an 8-bit encoded voltage level. A RTL simulation of this module was carried out, and his behavior as shown in figure 5.21 where two addresses

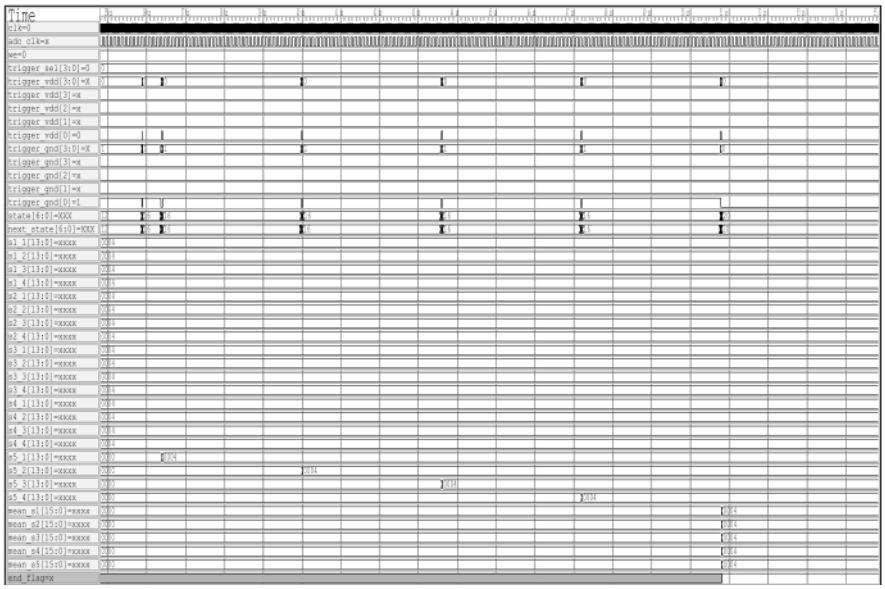


Figure 5.14: Sampling *s5* and activation of the *end_flag* signal

and two voltage levels are introduced at different times. Demonstrating in this way the correct operation of this module.

5.3.5 MUX module

The test system has an analog multiplexer for selecting between 8 input signals (four TIAs circuits and four integrators) to be sampled by the digital analog converter. This mux has 3 control pins, which allow you to select the signal that will be connected to the ADC.

Due to limitations of configurable outputs (FPGA ports) there is no module dedicated to control the MUX of the PCB, however it is possible to control it manually by stimulating its connections. The control of this module is explained in the next section.

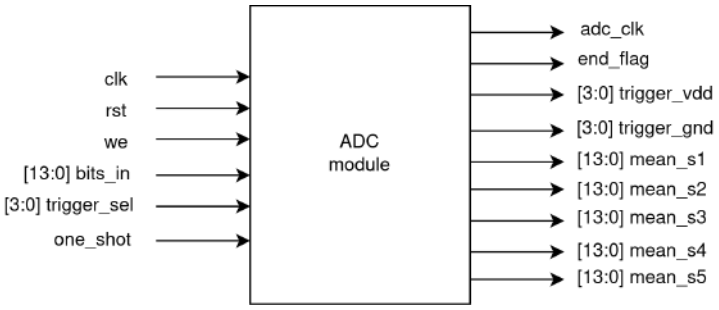


Figure 5.15: Block diagram of the ADC module

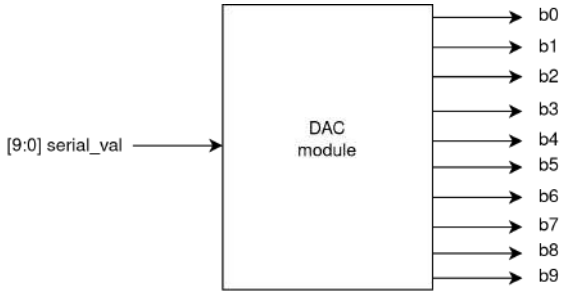


Figure 5.16: Block diagram of the DAC module

5.4 Documentation and examples of use of the AXI-IP's

Each of the hardware blocks described in the previous chapter were packaged as an AXI-IP, the latter in order to achieve communication between the RTL designs of the previous section and the ARM A9 processor, which is part of the SoC of the Zynq7000 series. The above allows that through software loaded in the ARM processor, registers of the packaged hardware can be written and read, thus allowing communication between the software and the designed hardware. In addition, this provides great versatility from the point of view of the flexibility of the use of the designed platform, allowing the

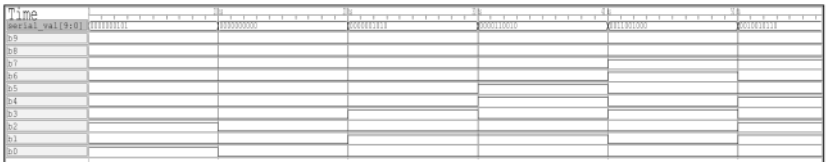


Figure 5.17: RTL simulation of the DAC module

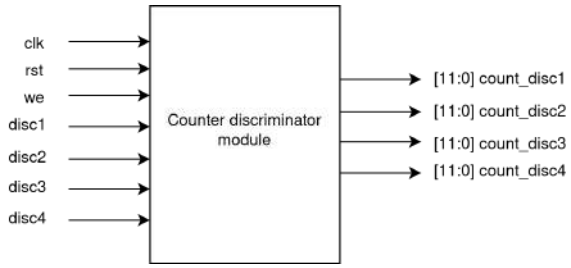


Figure 5.18: Block diagram of the DAC counter-discriminator module

user to write or read registers at will, in a controlled manner, using only the software and programming in C language. The interaction between software and hardware is based on memory addressing, where each hardware module has its memory addresses clearly defined. Through the use of pointers we can access these memory addresses, and thus be able to read or write them depending on the capabilities of each module.

This section provides the documentation for each of the AXI-IPs designed, with their respective examples of use. In the software there are pointers defined pointing to the base addresses of each of the modules. The values presented in the following tables are relative to these pointers and indicate N positions above the base addresses. The names of the pointers defined in the code are:

- ADC module: p_adc
- DAC module: p_dac

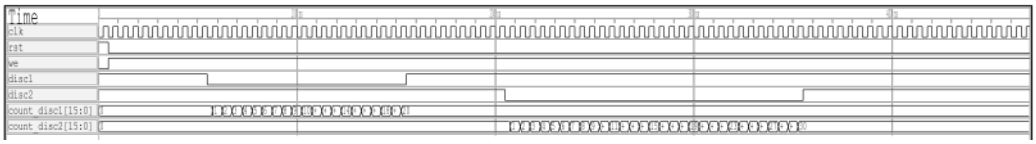


Figure 5.19: RTL simulation of the counter-discriminator module

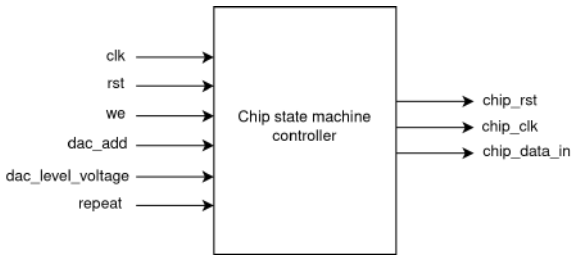


Figure 5.20: Block diagram of the chip state machine module

- Chip state machine controller module: *p_chip_fsm*
- Discriminator counter module: *p_disc*

Due to hardware limitations, control of this MUX will be done manually using some jumper pins of the PCB.

5.4.1 ADC Module

Registers description

Reset: resets all the internal values of the module, it is recommended to do a reset before any measurement or set of measurements.

Write enable: Enables or disables the writing of the module registers.

Trigger selector: Allows you to select which of the four trigger circuits will be used. For example, if the value written to this register is 0, trigger

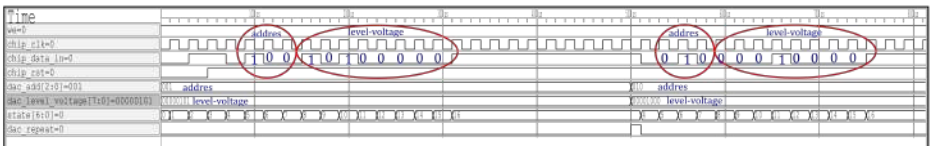


Figure 5.21: RTL simulation of the chip state machine module

Table 5.1: ADC module registers

Name	Bitwidth	Read or write	N (p_adc + N)
Reset	1	Write	0
Write enable	1	Write	1
Trigger selector	3	Write	2
One shot mode	1	Write	3
End flag	1	Read	4
Mean sample 1	16	Read	5
Mean sample 2	16	Read	6
Mean sample 3	16	Read	7
Mean sample 4	16	Read	8
Mean sample 5	16	Read	9
Output enable	1	Write	10

circuit number 1 is selected, if the value written to this register is 1, trigger circuit number 1 will be selected, and so on. (it is only valid to write up to 3, where the selected circuit will be number 4)

One shot mode: Set the module to work in single shot mode. This mode is useful when you want to take measurements of the discriminator circuit. In high state the module will operate in single shot mode, if the state of this input is low the circuit will operate in multiple shot mode, thus allowing to obtain average values of the outputs of the integrator circuits or of the TIA circuits.

Mean sample 1: Average value of the measurements captured using the subsampling methodology. It corresponds to the first sample (S1) of figure

5.9c.

Mean sample 2: Average value of the measurements captured using the subsampling methodology. It corresponds to the second sample (S2) of figure 5.9c.

Mean sample 3: Average value of the measurements captured using the subsampling methodology. It corresponds to the third sample (S3) of figure 5.9c.

Mean sample 4: Average value of the measurements captured using the subsampling methodology. It corresponds to the fourth sample (S4) of figure 5.9c.

Mean sample 5: Average value of the measurements captured using the subsampling methodology. It corresponds to the fifth sample (S5) of figure 5.9c.

Output enable: Enables or disables the operation of the ADC of the board (not of the module). This ADC input on the PCB is active low.

Usage examples: Writing to the registers

For this example it is assumed that there is a pointer p_adc which points to the base address of the ADC module. In addition, Xilinx proprietary functions are used that allow writing to the registers in a simple way.

```
1 //write a values to the reset,
2 //write enable, trigger selector and one shot inputs
3
4 Xil_Out32(p_adc, 1); //write 1 to the reset input
5 Xil_Out32(p_adc, 0); //write 0 to the reset input
6
7 Xil_Out32(p_adc+1, 1); //write 1 to the write enable input
```

```
8
9  Xil_Out32(p_adc+2, 1); //select trigger circuit 2

10 Xil_Out32(p_adc+2, 0); //select trigger circuit 1
11 Xil_Out32(p_adc+2, 2); //select trigger circuit 3
12
13 Xil_Out32(p_dac+3, 1); //write 1 to the one shot input mode
14
```

Usage examples: Reading from registers

It is possible to read the output data from the ADC module and load it into variables in the software. Below are code examples. For this example it is assumed that there is a pointer p_adc which points to the base address of the ADC module and previously defined variables where the read data will be stored.

```
1 //Code example. How to read outputs of the ADC module.
2
3     end_falg = *(p_adc+4);         //read end flag state
4     mean_s1 = *(p_adc+5);         //read mean s1 value
5     mean_s3 = *(p_adc+7);         //read mean s3 value
6     mean_s3 = *(p_adc+9);         //read mean s5 value
7
```

5.4.2 DAC Module

Table 5.2: DAC module registers

Nombre	Ancho de bits	Read or write	N (p_dac + N)
Serial value	10	Write	0

Registers description

- Serial value: Corresponds to the value that will be entered by software. This value will be translated by the module to a parallel output which will allow to have a specific output voltage for the DAC. It is important to mention that the DAC has 4 channels, the selection of these channels must be done physically on the PCB using jumpers. The selected channel will be the one that will be written with the serial value entered by software. Table 5.3 shows the configuration values specific to the manufacturer of the DAC used on the PCB. To know the output voltage based on the serial value encoded in binary (D), equation 5.1 must be used, where $Vref$ will be 3.3v and Gain is configurable by means of jumper JP2 on the PCB.

$$V_{out} = V_{ref} \cdot \frac{D}{2^N} \cdot Gain \tag{5.1}$$

Manual control of WR, LDAC and CLR inputs

Due to hardware limitations, the control of the WR, LDAC and CLR inputs must be done manually through the J7 connection bank of the PCB, which is connected to the ports in the order indicated in the 5.2. Figure 5.22 shows the J7 connection bank, the polarity of the sources, and the uniqueness of the WR, LDAC, and CLR inputs. The function of these inputs are mentioned below.

- WR: Enables or disables the WR pin of the DAC on the PCB. This pin is active low and allows you to enable writing to the DAC registers and also allows you to update the output values. Table 5.3 shows more information about this.

- CLR: Enables or disables the CLR pin of the DAC on the PCB. This pin is active low and allows you to clear the DAC registers from the PCB. Table 5.3 shows more information about this.

- LDAC: Enables or disables the LDAC pin of the DAC on the PCB. This pin is active low and allows updating the values of the DAC registers on the PCB. Table 5.3 shows more information about this.

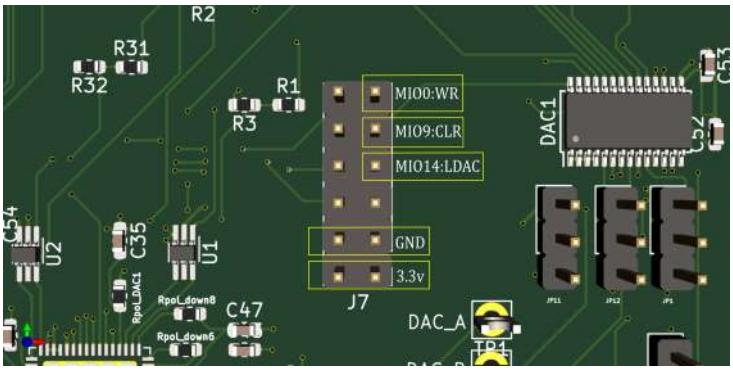


Figure 5.22: PCB J7 connection bank pin-out

Configuration jumpers

This section of the PCB (and the DAC module) has physical selection jumpers that allow you to configure some DAC options.

Previously we mentioned jumpers J11 and J12, which allow you to select the channels of the DAC. In addition to these last two jumpers, there are

two others, JP1 and JP2. The latter allow you to set the CS and Gain pins of the DAC on the PCB.

The behavior of the DAC with CS in high or low state is shown in Table 5.3 and for the Gain pin it is established that if Gain is in high state Gain will take the value 2 in the equation 5.1, if Gain is in low state the value in equation 5.1 will be 1. It is important to mention that each of the four DAC outputs is configurable and independent of the others.

The jumpers and their polarity are shown in the figure 5.23.

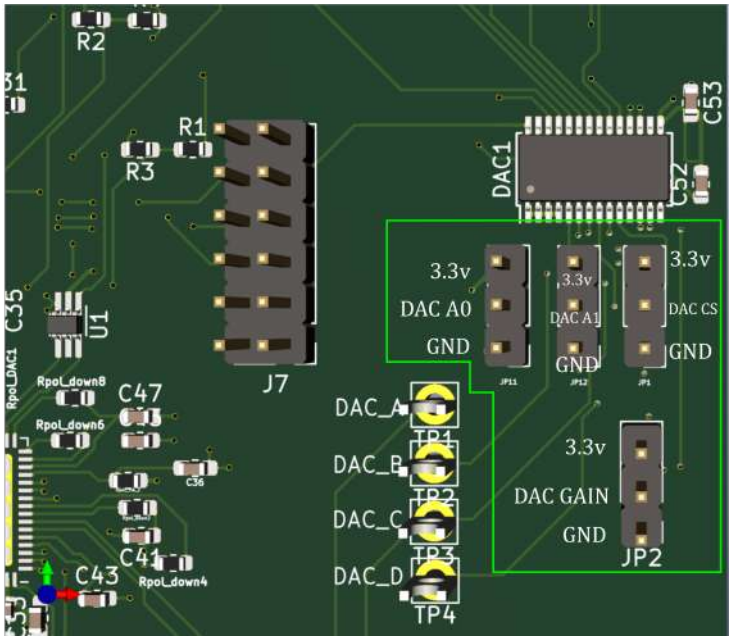


Figure 5.23: DAC jumpers, pin-out and their polarity

Usage example: Writing to the registers

For this example it is assumed that there is a pointer p_dac which points to the base address of the DAC module. In addition, Xilinx proprietary func-

Table 5.3: DAC configuration ports

∼CLR	∼LDAC	∼CS	∼WR	A1	A0	Function
1	1	1	X	X	X	No data transfer
1	1	X	1	X	X	No data transfer
0	X	X	X	X	X	Clear all registers
1	1	0	0 to 1	0	0	Load DAC A Input Register, GAIN A
1	1	0	0 to 1	0	1	Load DAC B Input Register, GAIN B
1	1	0	0 to 1	1	0	Load DAC C Input Register, GAIN C
1	1	0	0 to 1	1	1	Load DAC D Input Register, GAIN D
1	0	X	X	X	X	Update DAC Registers

tions are used that allow writing to the registers in a simple way.

```
1 //write serial_val
2 //serial_val can be any number between 0 to 2^10 - 1.
3
4 Xil_Out32(p_dac, serial_val);
5 Xil_Out32(p_dac, 333); //write 333 to the serial value input
```

The LDAC, WR and CLR inputs can be set low or high by connecting to the corresponding J7 $3.3V$ or GND connector pins respectively. It is also important to consider that the selection of the DAC channel to write is manually selected using jumpers J11 and J12 as mentioned above. Suppose then that you want to write a new value to the second DAC, the steps to follow according to the Table 5.3 are:

1. Connect the CLR pin to GND

2. Disconnect the CLR pin from GND and connect it to 3.3V

3. Connect LDAC to 3.3V, CS (jumper JP1) to GND and WR to GND
4. Connect jumper J11 to GND and J12 to 3.3V
5. Disconnect WR from GND and connect to 3.3V
6. Connect CLR to 3.3V and LDAC to GND

With the previous steps the value written in the variable *serial_val* will be loaded to the selected channel of the DAC. If you want to load a new *serial_val* value to any of the DAC channels, you must update the variable in the software and repeat the previous steps.

If the previous steps are not to be carried out manually, it is possible to connect the outputs of some microprocessor to the connection bank J7.

Usage example: Reading from registers

This particular module does not have internal registers that can be read, however the PCB has test points where it is possible to measure the output voltages of each DAC channel.

5.4.3 Chip state machine controller

Table 5.4: Chip state machine controller registers

Name	Bitwidth	Read or write	N (p_chip_fsm + N)
Reset	1	Write	0
Write enable	1	Write	1
DAC address	3	Write	2
DAC value	8	Write	3
DAC repeat	1	Write	4

Registers description

- Reset: resets all the internal values of the module, it is recommended to do a reset before any measurement or measurement set.
- Write enable: Enables or disables the writing of the module registers.
- DAC address: Allows you to select one of the 4 DAC's of the chip to be configured.
- DAC value: 8-bit value that will be introduced to the DAC selected with DAC address, this value will translate into a voltage level between 0 and 1.8V. The formula to calculate the voltage value will be $V_{out} = V_{ref} \cdot \frac{D}{2^N}$ where V_{ref} is $1.8V$, D is the DAC serial value (between 0 and 2^{N-1}) and $N = 8$.
- DAC repeat: Allows you to reselect a new DAC address value to select and configure a new voltage value in the selected DAC. This input should normally be low and should only be high when you want to configure a new DAC and a new value for it, then it should return to a low state.

Usage example: Writing to the registers

For this example it is assumed that there is a pointer *p_chip_fsm* which points to the base address of the chip state machine controller module. In addition, Xilinx proprietary functions are used that allow writing to the registers in a simple way.

```
1 //write values to the reset, write enable, trigger selector
     and one shot inputs
```

```
2
3 Xil_Out32(p_chip_fsm, 1); //write 1 to the reset input (set
     high state)
4 Xil_Out32(p_chip_fsm, 0); //write 0 to the reset input (set
     low state)
5
6 //write 1 to the write enable register (set high state)
7 Xil_Out32(p_chip_fsm+1, 1);
8
9
10 //write the value two to the DAC address for select the DAC
     number 2
11 Xil_Out32(p_chip_fsm+2, 2);
12
13 //write the value three to the Dac address for select the DAC
      number 3
14 Xil_Out32(p_chip_fsm+2, 3); //the DAC 3 is selected
15
16 //write values to the Dac value register
17 Xil_Out32(p_chip_fsm+3, 77); //write 77 to the DAC value
     register
18 Xil_Out32(p_chip_fsm+3, 105); //write 105 to the DAC value
     register
19
20
21 //Code sequence to set a value to one of the four DACs on the
      chip
22
23 Xil_Out32(p_chip_fsm, 1); //reset the module
24
25 Xil_Out32(p_chip_fsm, 0); //reset OFF
```

```
26 Xil_Out32(p_chip_fsm+1, 1); //we enable writing
27 Xil_Out32(p_chip_fsm+4, 0); //dac repeat off
28
29 Xil_Out32(p_chip_fsm+2, 3); //the DAC 3 is selected
30 Xil_Out32(p_chip_fsm+3, 80); //write 80 to the Dac value
      register
31
32 //now if we would like to select another DAC of the chip and
      set a new value.
33
34 Xil_Out32(p_chip_fsm+2, 3); //set the new Dac address
35 Xil_Out32(p_chip_fsm+3, 80); //write a value to the new DAC (
      or write a new value    for the same last DAC)
36
37 Xil_Out32(p_chip_fsm+4, 1); //dac repeat on. Here the values
      will be update.
38
39 // put off repeat register until you want to change the
      address or the Dac value
40 Xil_Out32(p_chip_fsm+4, 0);
```

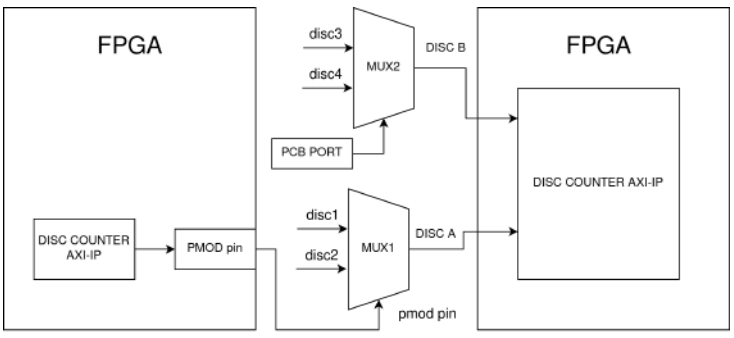
Usage example: Reading from registers

This particular module does not have internal registers that can be read, however the PCB has test points where it is possible to measure the output voltages of each of the four DACs on the chip.

5.4.4 Discriminator counter module

Due to hardware limitations, this module is divided into two sections. Both sections each depend on a 2 : 1 mux, which multiplexes the outputs of the discriminator circuits to two inputs of the FPGA. The control of each of the two 2 : 1 mux's is done for the first case using a jumper available on the PCB, with this it is possible to control the mux selector. The second section makes use of an AXI-IP with an output that allows to control the selector pin of the mux. Figure 5.24 shows graphically what was mentioned above, in this image you can see the control of MUX1 and MUX2 as well as the interconnection of the outputs of these multiplexers with the dedicated digital logic implemented in the FPGA.

The AXI-IP of the second mux also has two inputs, to which the outputs of the 2 : 1 mux's are connected. In this way, the processing of the discriminator circuits of the chip is carried out.



PCB PORT	DISC B
0	disc4
1	disc3

PMOD pin	DISC A
0	disc2
1	disc1

Figure 5.24: MUX1 and MUX2. Interconnection details

Table 5.5: Discriminator counter module registers

Name	Bitwidth	Read or write	N (p_adc + N)
Reset	1	Write	0
Write enable	1	Write	1
Counter Disc A	16	Read	2
Counter Disc B	16	Read	3
Select Disc A	1	Write	4

Table 5.6: MUX manually configration info

Name	Bitwidth	Read or write
MUX 2 select	**1**	**Write**

Registers description

- Reset: resets all the internal values of the module, it is recommended to do a reset before any measurement or set of measurements.
- Write enable: Enables or disables the writing of the module registers.
- Counter Disc A: This register contains the number of clock cycles that the output of the selected discriminator ($disc1$ or $disc2$) was in a low state. This value has a resolution of $5ns$, so the value contained in this register can be converted to time units considering the $1 : 5ns$ ratio.
- Counter Disc B: This register contains the number of clock cycles that the output of the selected discriminator (disc3 or disc4) was in a low state. This value has a resolution of $5ns$, so the value contained in this register can be converted to time units considering the $1 : 5ns$ ratio.
- Select Disc A: This register controls the selector pin of MUX1 (Figure

5.24.

Usage example: Writing to the registers

For this example it is assumed that there is a pointer p_disc which points to the base address of the module. In addition, Xilinx proprietary functions are used that allow writing to the registers in a simple way.

```
1
2 Xil_Out32(p_disc, 1); //write 1 to the reset input
3 Xil_Out32(p_disc, 0); //write 0 to the reset input
4
5 Xil_Out32(p_disc+4, 0); //write 0 to the Select Disc A
6 Xil_Out32(p_disc+4, 1); //write 1 to the Select Disc A
```

In the case of MUX2, the selector pin is controlled through the use of one of the pins connected to the connection bank $J7$ of the PCB, which implies that the selection must be made manually, connecting the pin to a high state ($3.3V$) or to a low state (GND). The location of this pin is shown in Figure 5.25.

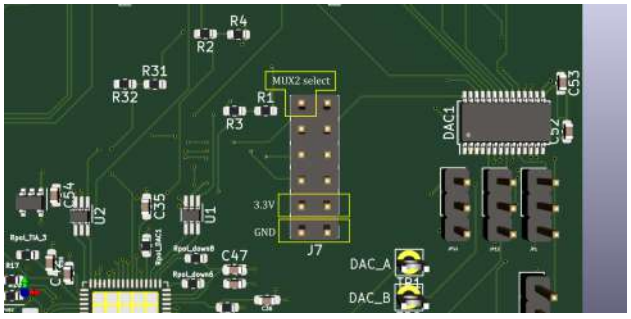


Figure 5.25: MUX2 selector pin. Location on the J7 jumper

Usage example: Reading from registers

It is possible to read the output data from the module (counter disc A, counter disc B) and load it into variables in the software. Below are code examples.

For this example it is assumed that there is a pointer *p_disc* which points to the base address of the module and previously defined variables where the read data will be stored.

```
1 //Code example. How to read outputs of de ADC module.
2
3  counter_disc_a = *(p_disc+2);        //read value of the
     counter A
4  counter_disc_b = *(p_disc+3);        //read value of the
     counter B
```

5.4.5 MUX8:1 control

The mux has 3 control pins, A, B, C and in the Table 5.7 is the output logic for multiplexing.

Table 5.7: MUX8:1 configuration table

A	B	C	OUT
0	0	0	X0
0	0	1	X1
0	1	0	X2
0	1	1	X3
1	0	0	X4
1	0	1	X5
1	1	0	X6
1	1	1	X7

Table 5.8: MUX8:1 module registers

Name	Bitwidth	Read or write
A	1	Write
B	1	Write
C	1	Write

Description of the variables of the Table 5.8:

- A: Multiplexer input
- B: Multiplexer input
- C: Multiplexer input
- Out: Multiplexed output

Instructions for use

The multiplexer configuration must be done manually, by pinning inputs A, B and C high or low as required and based on the specifications in Table 5.7. These MUX8:1 inputs are connected to the $J7$ connection bank of the MUX8:1. PCB and in Figure 5.26 the physical location of these pins are indicated.

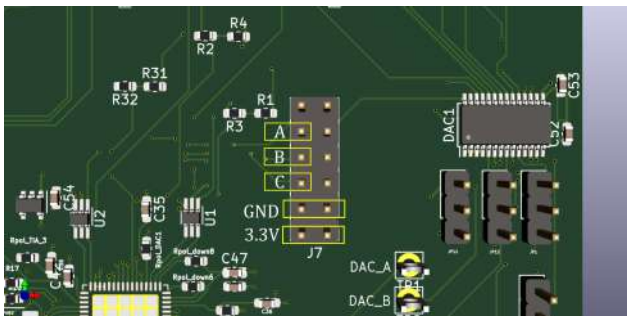


Figure 5.26: MUX8:1 assigned pins locations

6

Integration of the AXI-IPs to the Zynq7000 SoC

To make a flexible characterization platform for the chip, will be great if the IP modules could be modified using firmware. So, the IPs described and documented in the last chapter need to be integrated with the ARM processor of the Zync7000 SoC, this would possible if the IP modules incorporate protocol communication that permits communication between the IPs and the ARM processor. This was done by incorporating the IP modules into modules predesigned by Xilinx (wrapping process), which incorporate the AXI communication interface. The ARM processor also includes this communication interface, thus allowing communication between the IPs and the processor.

To comply with the above, the synthesis and implementation stages must be completed in Vivado, without errors and complying with the timing constraints for both design stages. In this particular case, the FPGA logic must operate in $5ns$ periods, this being one of the most important design constraints, because the entire system was designed to operate in single-cycle mode, that is, all operations must be performed in one clock cycle at most.

6.1 Synthesis and implementation in the FPGA

The synthesis process maps the RTL designs to elements called standard cells available in the FPGA hardware, thus providing data on physical characteristics (area, delays, power consumption, etc.) which allows estimates and a first approach to real design.

The implementation guarantees that the designed circuits can be executed in the FPGA and that there are no inconsistencies between the synthesized circuit and the functional logic that can be implemented with the available resources of the FPGA. In addition to this, the synthesis, implementation and bitstream generation reports must be verified and ensure that there are no errors or critical warnings that are related to timing or interconnections of the designed logic to GND or VDD permanently. The latter would result in the impossibility of modifying said inputs or outputs through software due to an implementation impediment of the synthesized circuit.

The implemented system is presented in the Figure 6.1 where it is possible to observe the ARM processor(Zynq processor system), the incorporation of the AXI interconnections, and the AXI-IPs designed with their respective outputs to the outside world (physical pins of the FPGA).

The design presented in the Figure 6.1 managed to be synthesized and implemented in the FPGA, without errors, and meeting the necessary requirements to operate and control the chip test environment. However, there were some problems in the implementation stage that are important to mention/document, these problems are presented below.

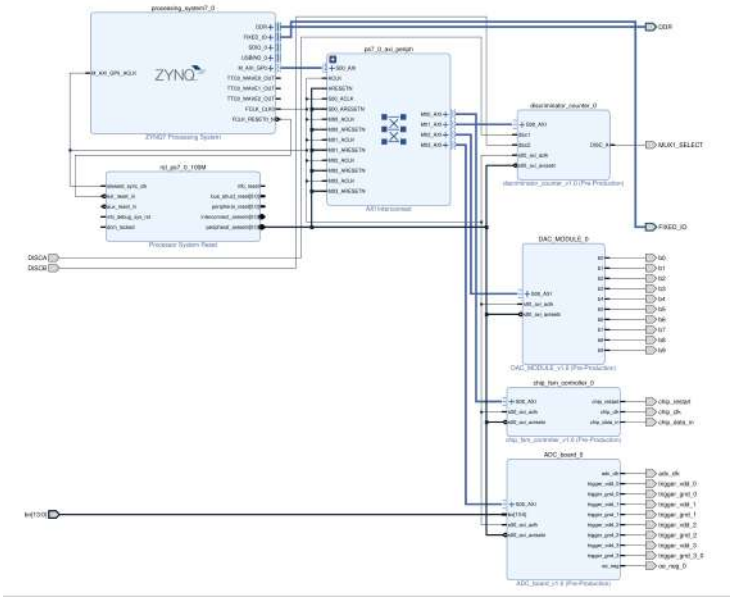


Figure 6.1: IP modules integrated with the Zynq processor system

6.1.1 Problems during implementation

After the synthesis of the circuits shown in the Figure 6.1 , the implementation of the circuits on the FPGA must be carried out. In this stage, hardware optimizations had to be made due to timing errors related to the discriminator-counter module. The problems were related to hold errors and the solution was to optimize the internal connections of the module, reducing the width in bits of the registers and cables.

On the other hand, the state machine in charge of controlling the state machine of the chip was coded in such a way that the synthesized circuit, when implemented, kept the outputs of the state machine permanently connected to ground. This error was due to an error in the RTL encoding (double assignment of the outputs in different always blocks). The rest of the blocks presented no problems.

After the solution of the implementation errors, these modules were verified. The results are presented in the next chapter.

7 Results of the system implementation

This chapter presents the corroboration of the correct operation of the hardware modules described in the previous chapters, thus allowing to validate the operation of the test environment and the possibility of reconfiguration based on the requirements of the experiments.

7.1 Verification and results of software-hardware to real world interaction

To carry out the verification, a logic analyzer will be used, which will be connected to the FPGA pins, allowing the behavior of the outputs to be observed and the software-hardware-real world integration to be validated. Remember that the FPGA will be connected to the PCB through the input/output banks JA, JB, JC, JD, and JE. The pin assignment is presented in Table 8.1 and in Figure 7.1 you can see the FPGA with the respective input and output banks mentioned above.

The results presented in this section focus on the verification of the environment designed in the FPGA, assuming that the PCB is not re-configurable and that it will respond correctly to the stimuli delivered by the FPGA. The results are concentrated in the verification of the generation of clocks (ADC

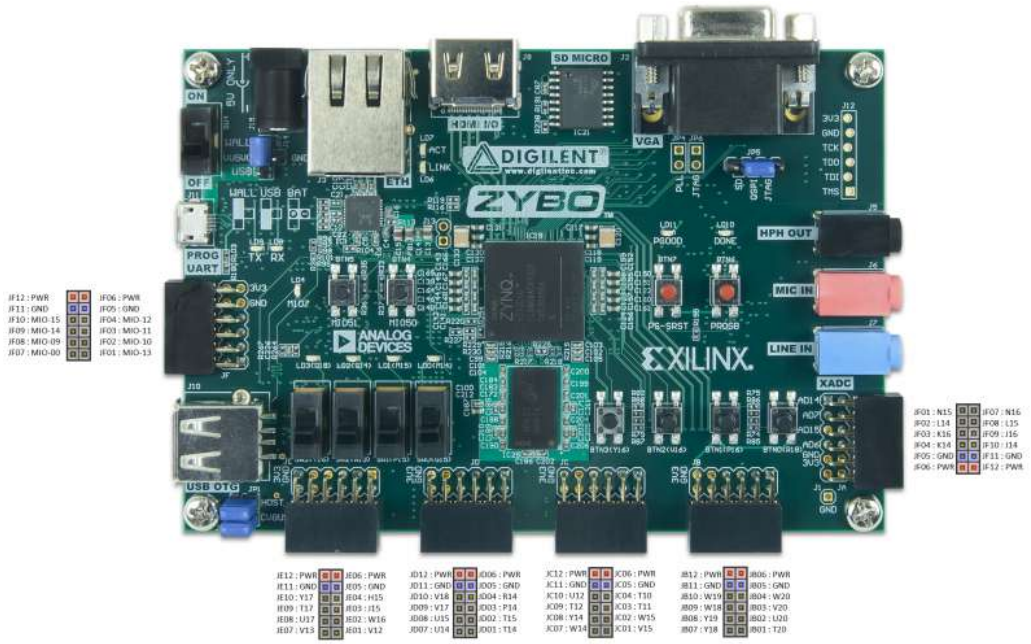


Figure 7.1: Zybo FPGA connector pins (from Zybo reference manual)

and state machine of the chip), writing to the DAC of the PCB and to the DACs of the chip by means of the control of the state machine that this one possesses.

In the annexes of this thesis, instructions for use and test algorithms are presented that can be used to carry out tests and characterizations of the chip using the environment designed in this work.

7.1.1 ADC module

The signal generation of the ADC module was verified using the logic analyzer. It is important to mention that in this test the system clock frequency was reduced (4 times), this is because the analyzer was not able to measure correctly at the original frequency (hardware limitations).

The test was configured in the software and programmed to the FPGA and the signals measured with the logic analyzer were: the ADC clock, one of four trigger signals that connect one of the four charge generators to GND, one of four trigger signals that connect one of the four charge generators to VDD and the end flag signal. The results and the measurements can be seen in Figure 7.2

Figure 7.2: ADC module, test results: Clock and signals generation

7.1.2 Reading the discriminator counter values

The values of the counter values register were sent through a serial interface and read using a USB-serial converter (available on the FPGA). Another signal was programmed on the FPGA for simulating the ON-OFF transition of the chip discriminator output, the duration of the transition (measured on microseconds) was defined in the software. Three values for ON-OFF transition time were tested: 0.2, 20, 32 (microseconds). The results of this experiment are presented in Figure 7.3, where the text output corresponds to data read directly from the FPGA USB-serial interface.

```
useconds disc1: 0.2 | clk cycles (T=5ns): 40
useconds disc2: 20 | clk cycles (T=5ns): 4000
useconds disc2: 32 | clk cycles (T=5ns): 6400
```

Figure 7.3: Discriminator counter module, test results: Reading the internal register values

7.1.3 MUX1 selector

The output that control the selector of the MUX1(see Figure 5.24) is controlled for the counter discriminator module. For testing this output signal the Zynq processor system was programmed whit a toggle routine. The results of this experiment can be seen in the Figure 7.4, where the signal of the MUX1 selector is changed to high state to low state (one second on, one second off) corroborating the correct behavior of the output signal and the possibility of using software for control this signal.

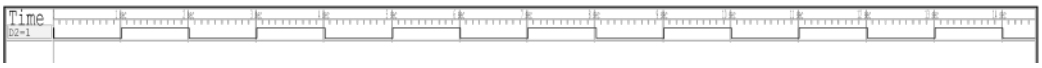


Figure 7.4: MUX1 selector, test results: Control and generation of the selector signal

7.1.4 DAC values

The DAC is a fully combinational module, but must be enabled and configured using software. To corroborate the correct operation of this module, different values were loaded through the software and the outputs corresponding to the DAC output bits were measured using the logic analyzer.

The measurements are presented in Figure 7.5, also in this figure you can see the values: 100, 2, 7 and 5, there values are represented on binary format (parallel digital outputs) and was introduced using C code.

7.1.5 Writing to on-chip DAC's

The DAC's on the chip are controlled by a state machine. To control this state machine, a state machine must be implemented in the FPGA for enable the

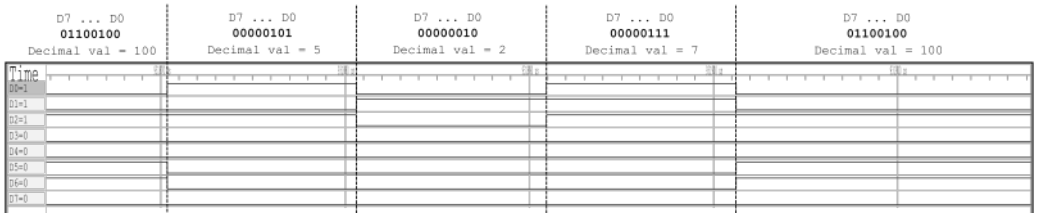


Figure 7.5: DAC module, test results: Hardware parallel encoding of the software values

control of the chip state machine. This state machine must comply with the logical sequence of selection and writing to the DAC's of the chip presented in the **Renzo thesis.**

The specifications for inputting information to the DAC's are:

1. The clock is operating.

2. rst = LOW . The state machine go into reset.

3. data_in = HIGH. The state machine enters the idle state.

4. rst = HIGH. The state machine is enabled to operate, but remains needs in idle state.

5. datain = LOW . On the next falling edge, the state machine will exit from the idle state and will go to the state in which it receives the address of the DAC that is written.

6. Through data in, send the desired DAC address serially to write. The shift-register registers each bit sent on the rising edges of the clock (least significant bit is sent first). The DAC address is described by a 3-bit word.

7. Using data in, send serially the digital value that you want to enter to the DAC indicated in the previous step. The shift-register registers each bit sent on the rising edges of the clock (the least significant bit is sent first).

8. data in = HIGH. This brings the state machine to its idle state.

9. To repeat the write, do data in = LOW . This will take us back to step 5 of this list of instructions.

To corroborate the correct generation of control signals for the chip state machine, various parameters were configured in the software (varying the selected dac and the voltage value) and the outputs were observed with the logic analyzer on the corresponding pins of the chip. FPGA. These results can be seen in Figure 7.6. The clock signal also can be seen in Figure 7.6.

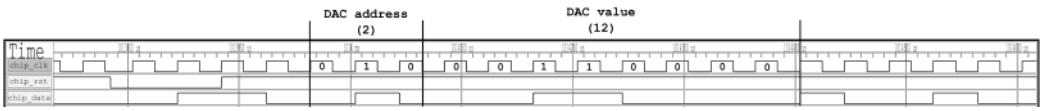


Figure 7.6: Chip state machine module, test results: Clock generation and test writing to the on chip DAC's

7.2 PCB manufacture

The PCB was sent to manufacture successfully, meeting the necessary dimensions to incorporate the FPGA correctly, presenting a solid and reconfigurable evidence base to the needs of the experiments. In the Figure 7.7 you can see the FPGA attached to the PCB through vertical connectors. This

connection facilitates the stimulation of the different components of the PCB by the FPGA, facilitating both the writing and reading of signals.

Figure 7.7: FPGA attached to the PCB through vertical connectors

8 Conclusions and future work

Based on the data provided by the chip designer in his Thesis (Altamirano, 2021), it is possible to use this information to devise a test scenario for the ASIC. The test environment must be adjusted to the physical requirements of the Chip for its correct functioning and operation.

The PCB to which the chip must be soldered has been designed and manufactured. Additional components were selected and justified based on the testing requirements of the chip and with the goal of having a flexible platform that could be reconfigured using an external programmable circuit, in this case an FPGA. The digital logic implemented in the FPGA was verified at the RTL level, and after the implementation in the FPGA. The correct operation of this implementation and the ability to be reconfigurable through software was corroborated by measuring the FPGA outputs with different programmed routines, where the measured outputs correspond to the expected behavior. With this, the correct programming of an external, reconfigurable circuit is concluded, capable of controlling the inputs and outputs of the chip when necessary. In addition, the ability of the system to record and extract data from the experiments using software and a usb-serial converter available in the FPGA has been confirmed. For the above, in the appendix of this thesis, examples and references to examples of use have

been left. The last 3 chapters of this thesis can be considered as a user manual for the system, with all the information and documentation necessary to understand the capabilities and characteristics of the designed testing environment. In addition, some examples and results of use are presented, as well as, in the appendix of this document there are explicit examples for the use of the system's functionalities.

From the work carried out in this thesis and the results obtained in the verification of the hardware implemented in the FPGA, the verification of the generation of signals capable of stimulating the PCB, the corroboration of the flexibility of the platform to adapt to the needs of users who will carry out chip characterization activities and the ability to modify the behavior of the system and obtain data from it in a simple way using software, we can conclude that the objective of designing and implementing a test environment that allows testing with the ASIC is has successfully achieved.

8.1 Future work

The chip must be soldered to the PCB and a series of experiments to be carried out with the platform must be defined. The user of the system must adjust it to the needs and requirements of the experiments that he wants to carry out. It is expected to obtain data on the performance of the PCB once the first experiments have been carried out, based on the performance of the system, the PCB could be updated to, for example, have more test points, parallelize measurements or vary the physical distribution of some components. The FPGA hardware could also be upgraded/modified in future releases if the needs of future experiments change. In addition, it would be

ideal to port the designed hardware to an FPGA that has a greater number of outputs, in order to avoid manual control of some of the PCB elements and fully automate the characterization process through the use of software, such as which as demonstrated for some of the AXI-IP's modules designed in this work.

Appendix

Below are some details and considerations that may be useful when using the platform designed in this work.

8.2 FPGA pin assignment

The Table 8.1 presents the FPGA pins assignments (PMOD connectors). This table will be of a great utility when the experiments with the PCB carry out.

8.3 PCB schematic

The Github repository mentioned above also contains the schematic files of the PCB. The project can be loaded revised using KiCad and there are some SVG files that contain the PCB board schematic view. Also, in Figure 8.1 it is possible to see the labeling interconnection and the distribution of the components on the PCB.

8.4 How to use the system: Details and examples

In general, the use of the system for do experiments with the chip will have the following logical sequence, divided into two sections: PCB configuration and FPGA programming.

PCB Configuration

- PCB power supply configuration.

Table 8.1: FPGA pins assignments(1)

FPGA	**PCB Function**
JE1:V12	ADC:D8
JE2:W16	ADC: D9
JE3:J15	ADC: D10
JE4:H15	ADC: D11
JE7:V13	ADC: D12
JE8:U17	ADC: D13
JE9:T17	ADC: 'OE
JE10:Y17	ADC:CLK
JD1:T14	ADC: D0
JD2:T15	ADC: D1
JD3:P14	ADC: D2
JD4:R14	ADC: D3
JD7:U14	ADC: D4
JD8:U15	ADC: D5
JD9:V17	ADC: D6
JD10:V18	ADC: D7
JC1:V15	DAC: DB8
JC2:W15	DAC: DB9
JC3:T11	DISCA
JC4:T10	DISCB
JC7:W14	CHIP: CLK
JC8:Y14	CHIP: RST
JC9:T12	CHIP: DATA_IN
JC10:U12	MUX2:1_1_SELECT

- PCB jumper configuration (ADC, DAC, VNC's, trigger system).
- Configuration of the values of the on chip DAC's.
- Configuration of the PCB MUX's for reading the data of the ADC and the discriminator counter module.

Program and use the FPGA using Vivado

- Selection of the charge generator circuit (using the ADC module).
- Configuration of the ADC module for single-shot or multi-shot mode.
- Configuration of the charge generator circuit voltage (using the DAC module).

Table 8.2: FPGA pins assignments(2)

FPGA	PCB Function
JB1:T20	DAC: B0
JB2:U20	DAC: B1
JB3:V20	DAC: B2
JB4:W20	DAC: B3
JB7:Y18	DAC: B4
JB8:Y19	DAC: B5
JB9:W18	DAC: B6
JB10:W19	DAC: B7
JA1:N15	CHGEN1:1S vcc0
JA2:L14	CHGEN1:2S gnd0
JA3:K16	CHGEN2:3S' vcc1
JA4:K14	CHGEN2:4S' gnd1
JA7:N16	CHGEN3:5S vcc2
JA8:L15	CHGEN3:6S gnd2
JA9:J16	CHGEN4:7S vcc3
JA10:J14	CHGEN4:7S gnd3
JF1:MIO13	MUX2-1_2_SELECT(D)
JF2:MIO10	MUX8-1_A
JF3:MIO11	MUX8-1_B
JF4:MIO12	MUX8-1_C
JF7:MIO0	DAC: WR'
JF8:MIO9	DAC: CLR'
JF9:MIO14	DAC: LDAC'
JF10:MIO15	Not Connect

- Trigger charge generator circuit
- Extract the data measured by the FPGA, which are stored in the internal registers of the ADC module and the discriminator counter module.

8.4.1 Details about FPGA programming

The digital logic described in the previous chapters is ready and available to be loaded to the FPGA in a simple way using Vivado and the *chip_test_env_wrapper.xsa* hardware configuration file, this file contains all the hardware configuration described in the previous chapters (AXI- IP's). In the Github repository, a default project is available and ready to be opened in Vivado, this project

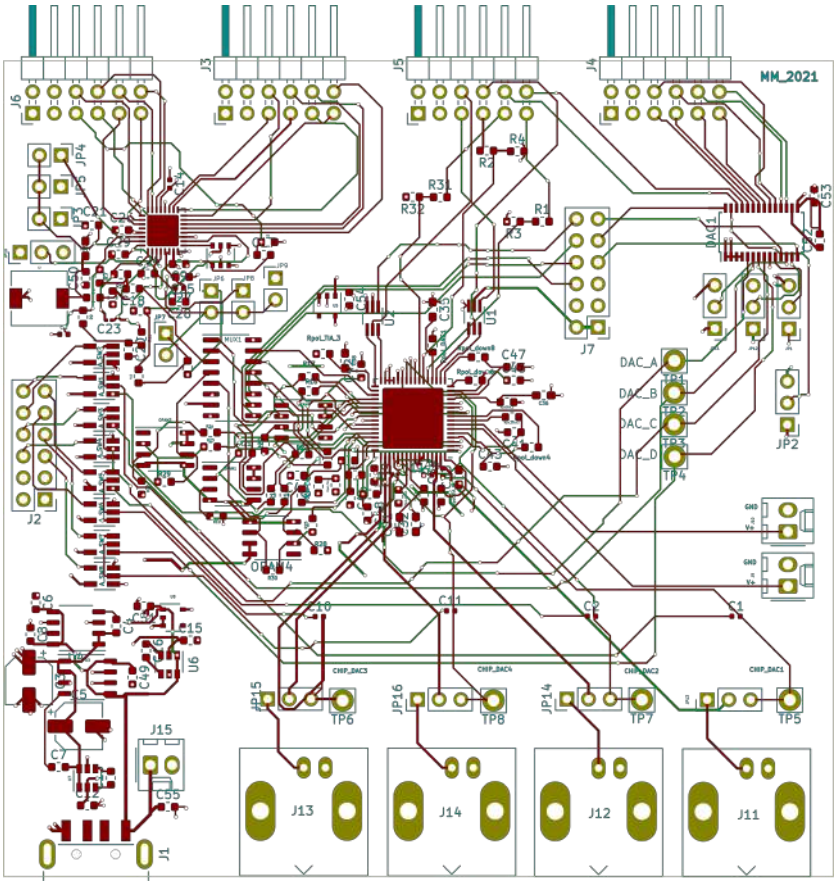


Figure 8.1: PCB schematic: interconnection and the distribution of the components on the PCB

contains everything necessary to only concentrate on configuring the software that will be uploaded to the SoC's ARM processor. The files of this project also include example codes for the use of AXI-IP's, which can be modified according to the user's needs. Additionally, some of those examples are shown below.

How to write values to on chip DAC's

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xparameters.h"
#include "xil_types.h"
#include "xil_io.h"
```

```

int *p_chip_fsm;
int main() {
    //-- Custom AXI-IPs base address declaration BEGIN --//
  init_platform();  //init uart communication
    p_chip_fsm = XPAR_CHIP_FSM_CONTROLLER_0_S00_AXI_BASEADDR;

     //-- Custom AXI-IPs base address declaration END --//

   Xil_Out32(p_chip_fsm, 1); //reset the module
   Xil_Out32(p_chip_fsm, 0); //reset the module
   Xil_Out32(p_chip_fsm+1, 1); //we enable writing

   Xil_Out32(p_chip_fsm+2, 3); //the DAC 3 is selected
   Xil_Out32(p_chip_fsm+3, 14); //write 80 to the Dac value registe
   Xil_Out32(p_chip_fsm+4, 1); //write 80 to the Dac value registe

   Xil_Out32(p_chip_fsm, 1); //reset the module
   Xil_Out32(p_chip_fsm, 0); //reset the module

  while(1){
     Xil_Out32(p_chip_fsm, 1); //reset the module
     Xil_Out32(p_chip_fsm, 0); //reset the module
     Xil_Out32(p_chip_fsm+4, 1);
     Xil_Out32(p_chip_fsm+2, 2); //the DAC 3 is selected
     Xil_Out32(p_chip_fsm+3, 80); //write 80 to the Dac value register
  }
}
```

ADC module configuration example: Generate signals and take measurements

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xparameters.h"
#include "xil_types.h"
#include "xil_io.h"

int *p_adc;
int main()
{
    // ----- Custom AXI-IPs base address declaration BEGIN ----- //
  init_platform();  //init uart communication
```

```
13     p_adc = XPAR_ADC_MODULE_0_S00_AXI_BASEADDR;
14      // ----- Custom AXI-IPs base address declaration END
       ----- //
15
16   // ---- CODE EXAMPLE ----//
17
18   Xil_Out32 ( p_adc , 1) ; // write 1 to the reset input
19
20   Xil_Out32 ( p_adc +1 , 1) ; // write 1 to the write enable
      input
21
22   //selecting different trigger circuits
23   Xil_Out32 ( p_adc +2 , 1) ; // select trigger circuit 2
24   Xil_Out32 ( p_adc +2 , 0) ; // select trigger circuit 1
25   Xil_Out32 ( p_adc +2 , 2) ; // select trigger circuit 3
26
27   Xil_Out32 ( p_adc , 0) ; // write 0 to the reset input.
      System Start!
28
29   // Code example . How to read outputs of the ADC module .
30
31   end_falg = *( p_adc +4) ; // read end flag state
32   mean_s1 = *( p_adc +5) ; // read mean s1 value
33   mean_s3 = *( p_adc +7) ; // read mean s3 value
34   mean_s5 = *( p_adc +9) ; // read mean s5 value
35
36   while(1){
37   //send mean_s1, mean_s3, mean_s5 trought the serial port
38   Xil_printf("mean_s1 = %f | mean_s3 = %f, mean_s5 = %f",
      mean_s1, mean_s3, mean_s5);
39   }
40  }
```

PCB DAC configuration example: setting different voltage output levels

```
1  #include <stdio.h>
2  #include "platform.h"
3  #include "xil_printf.h"
4  #include "xparameters.h"
5  #include "xil_types.h"
6  #include "xil_io.h"
7
8   int *p_dac;
9  // write serial_val
10 // serial_val can be any number between 0 to 2^10 - 1.
11
12 int main()
```

```
{
    // ----- Custom AXI-IPs base address declaration BEGIN -----
  init_platform();  //init uart communication
    p_dac = XPAR_DAC_MODULE_0_S00_AXI_BASEADDR;

     // ----- Custom AXI-IPs base address declaration END -----

  Xil_Out32 ( p_dac , 100 ); // write 100 to the serial value input
  Xil_Out32 ( p_dac , 333); // write 333 to the serial value input
  Xil_Out32 ( p_dac , 1200 ); // write 1200 to the serial value input

  while(1){
    //loop
  }
   }
```

8.4.2 Configuration jumpers

The PCB has some jumpers that allow you to configure the behavior of the PCB and some of its elements. The configurations will be made based on the needs of the user, details of the configuration jumpers available on the PCB are given below.

DAC jumpers

The DAC on the pcb has one bank of jumpers, these jumpers have the function of controlling which of the four DAC outputs will be used and also one of these jumpers allows you to configure the gain of any of the four outputs, in Figure 8.2 is shown the schematic of these jumpers.

ADC jumpers

The ADC PCB has two banks of configuration jumpers. One of these banks is dedicated to the control of the input value of the sense pin of the ADC, in Figure 8.3 a schematic of these jumpers is shown. According to the ADC datasheet, we can mention the following information about the pin sense:

SENSE (Pin 30): Reference Programming Pin. Connecting SENSE to VCM selects the internal reference and a ±0.5V input range. VDD selects the internal reference and a ±1V input range. An external reference greater

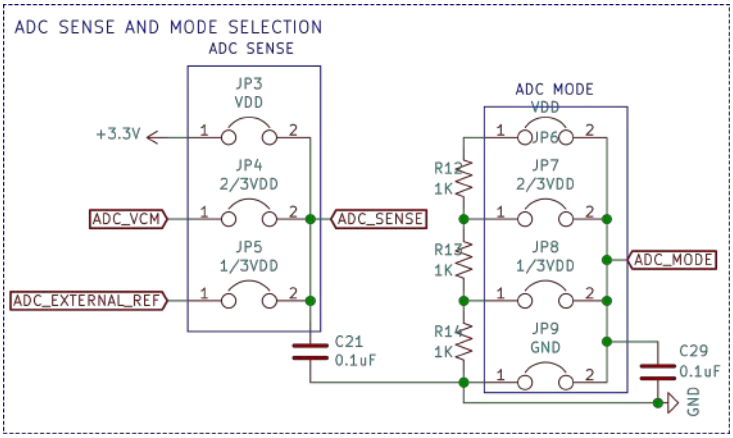


Figure 8.2: DAC configuration jumpers

than 0.5V and less than 1V applied to SENSE selects an input range of ±VSENSE. ±1V is the largest valid input range. The other of the configuration banks allows to control the operation mode of the ADC, Table 8.3 shows the available configurations of the ADC.

Table 8.3: ADC mode configuration

Mode Pin	Output Format	Clock Duty Cycle Stablizer
0	Offset Binary	Off
1/3Vdd	Offset Binary	On
2/3Vdd	2's Complement	On
Vdd	2's Complement	Off

Jumpers to control the selection of the charge injection source to the chip

The charge that stimulates the chip can come directly from the SiMP detectors through the VNC connectors arranged on the PCB (J11, J12, J13 and J14) or using the charge generator circuits arranged on the PCB. (both options should not be used at the same time). Figure 8.4 shows the schematic of the source selection jumpers for charge injection, where if the $350pF$ option is selected the load will be injected using one of the generator circuits. If the "detector" option is selected, the load will come directly from an external SiMP detector which will be connected to the corresponding VNC connector.

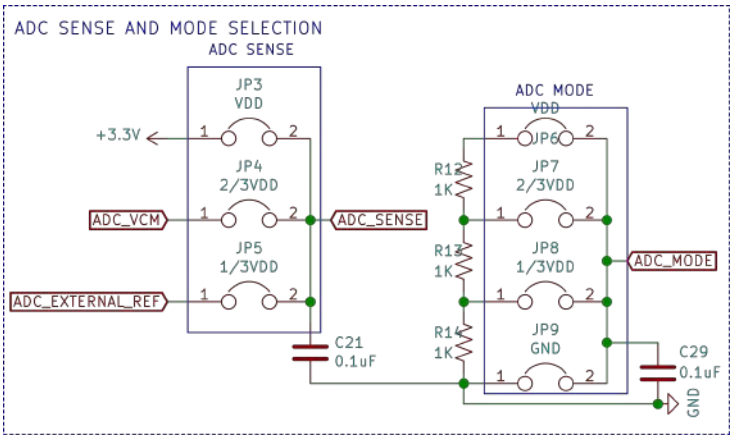


Figure 8.3: ADC configuration jumpers

If the selected option is to use the charge generator circuits available on the PCB, you must select which of the 4 modules to use and, on the other hand, the DAC value on the PCB must be set for that module. The above implies configuring the AXI-IP's of the ADC and the DAC for said purpose, that is to say that the AXI-IP of the ADC must be selecting one of the 4 discriminator modules and the DAC output must be set to the desired voltage value to inject a charge proportional to said voltage.

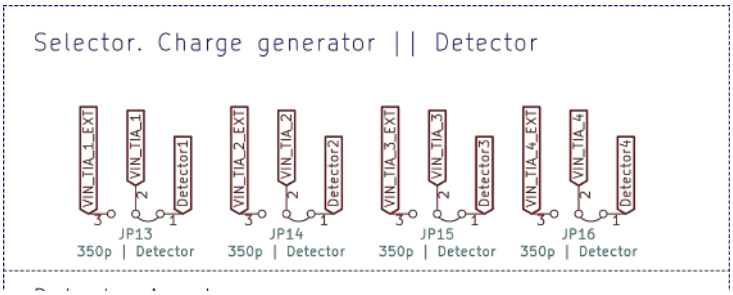


Figure 8.4: Selection jumpers for load injection

8.4.3 Some considerations about system power supplies

There are two power supplies in the system, these being: Analog and digital. The following describes in detail how to configure these power sources.

Analog power sources:

Analog power can be supplied from an external source (connector J15), it is also possible to use a USB connector to power the PCB. In both cases the input voltage cannot exceed 5.5V and only one of the two forms of power should be used at a time (negative voltages are not allowed) After being powered, from the J15 pin or from the USB connector, the power supply of the rest of the circuits will be controlled by a series of integrated circuits in charge of delivering the corresponding voltage levels to each of the elements of the PCB. All the analog power sources can be seen in Figure 8.5 (schematic view).

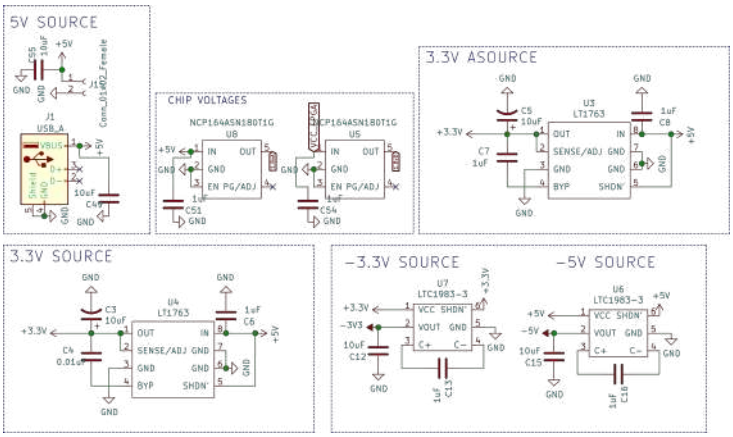


Figure 8.5: PCB analog power sources

Digital power source:

The digital supply comes directly from the FPGA through the PMOD pins, this supply has a value of 3.3V (VCC) and will energize all the digital logic of the FPGA. Figure 8.6 shows the pinout of the FPGA PMOD connectors.

Chip power supplies:

The chip has two power supplies (discriminator reference voltage and Vadj_inv), these voltages can be supplied through connectors J9 and J10 of the PCB respectively. Figure 8.7 shows the schematic of the J9 and J10 PCB connectors.

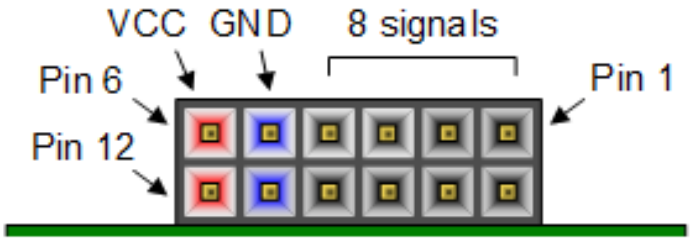


Figure 16. Pmod diagram.

Figure 8.6: Pinout of the FPGA PMOD connectors (from Zybo reference manual)

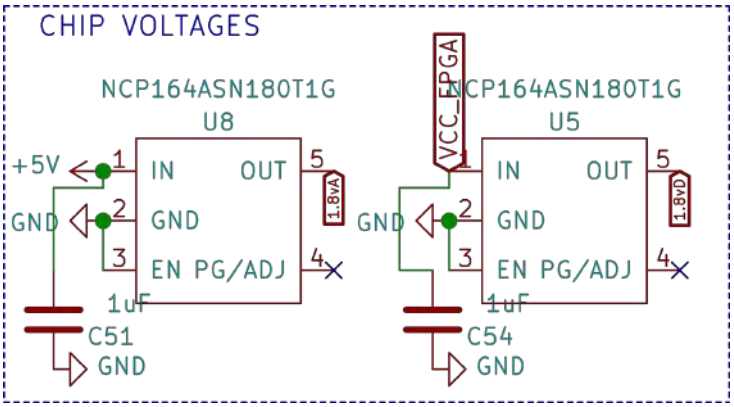


Figure 8.7: Chip power sources: J9 and J10 connectors

8.4.4 Components specifications

A list of the components(and the datasheets links) can be find on the Github reposotory.

References

Renzo Barraza Altamirano. Asic multicanal con control de ganancia para la lectura de detectores sipm, 2021.

MG Bagliesi, C Avanzini, G Bigongiari, R Cecchi, MY Kim, P Maestro, PS Marrocchesi, and F Morsani. A custom front-end asic for the readout and timing of 64 sipm photosensors. *Nuclear Physics B-Proceedings Supplements*, 215(1):344–348, 2011.

Pablo Walker Galdames. Slice-based analog design and its application to particle physics instrumentation, 2021.

Georges Gielen, Nektar Xama, Karthik Ganesan, and Subhasish Mitra. Review of methodologies for pre-and post-silicon analog verification in mixed-signal socs. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1006–1009. IEEE, 2019.

Xiaolong Guo, Raj Gautam Dutta, Yier Jin, Farimah Farahmandi, and Prabhat Mishra. Pre-silicon security verification and validation: A formal perspective. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

Technical note MPPC. HAMAMATSU, 2021.

Li-Jia Lin. *Improved Dual Phase Cross-Coupled Charge Pump Techniques for Selectable Multi-Output DC-DC Converters*. PhD thesis, 2012a.

High Voltage, Low Quiescent Current Inverting Charge Pump. Linear Technology, 2012b.

Dr. Ing. Ariel Lutenberg. *Analog to digital converters*. Universidad de Buenos Aires, 2012.

Jacob Millman and Christos C Halkias. Integrated electronics: analog and digital circuits and systems. *Tata McGraw-Hill Education: New Delhi*, 44: 45, 1972.

Subhasish Mitra, Sanjit A Seshia, and Nicola Nicolici. Post-silicon validation opportunities, challenges and recent advances. In *Design Automation Conference*, pages 12–17. IEEE, 2010.

Amir Nahir, Avi Ziv, Miron Abramovici, Albert Camilleri, Rajesh Galivanche, Bob Bentley, Harry Foster, Alan Hu, Valeria Bertacco, and Shakti Kapoor. Bridging pre-silicon verification and post-silicon validation. In *Design Automation Conference*, pages 94–95. IEEE, 2010.

C-Series SiMP Sensors. ON Semiconductors, 5 2021. Rev. 7.

Tadashi Orita, Mizuki Uenomachi, and Kenji Shimazoe. Development of time-over-threshold asics for radiation sensors.

Gaetano Palumbo and Domenico Pappalardo. Charge pump circuits: An overview on design strategies and topologies. *IEEE Circuits and Systems Magazine*, 10(1):31–45, 2010.

Sanjay Pithadia and Shridhar More. Grounding in mixed-signal systems demystified. 2013.

Behzad Razavi. *Design of analog CMOS integrated circuits.* , 2005.

Z Sadygov, A Olshevski, I Chirikov, I Zheleznykh, and A Novikov. Three advanced designs of micro-pixel avalanche photodiodes: Their present status, maximum possibilities and limitations. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 567(1):70–73, 2006.

Toru Tanzawa and Tomoharu Tanaka. A dynamic analysis of the dickson charge pump circuit. *IEEE Journal of solid-state circuits*, 32(8):1231–1240, 1997.

John P Uyemura. *Circuit design for CMOS VLSI*. Springer Science & Business Media, 2012.