UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Sviluppo del firmware per il controllo del detector ALPIDE ed implementazione su FPGA

Relatore

Prof. Gianmaria Collazuol

Laureando

Gabriele Bortolato

Mat. 1122493

Anno Accademico 2019/2020

Indice

Introduzione

Solid state sensors have become more and more central in high energy physics' detectors, thanks to their small size and low power consumption. In this work it is studied the ALPIDE sensor (ALice PIxel DEtector) chosen by the ALICE collaboration for the ITS upgrade (Inner Tracking System).
This sensor is a MAPS (Monolithic Active Pixel Sensor) it means that both the sensible device and front-end electronics share the same silicon die, this is necessary due to the stringent requirements set by ALICE, first of all low power density and high spatial resolution.
This work will be focussed on the developement of a firmware that will be implemented on an FPGA based board, it will control the ALPIDE chip and it will offer a simple comunication with PCs for data acquisition and testing purposes. The new readout and control system's performaces will be evaluated.

I sensori a stato solido giocano un ruolo sempre più fondamentale nello sviluppo di detector per la fisica delle alte energie, questo grazie alla loro compattezza e bassa densità di potenza necessaria per il loro funzionamento. In questa tesi si è studiato il sensore ALPIDE (ALice PIxel DEtector) scelto dalla collaborazione ALICE per l'upgrade dell' ITS (Inner Tracking System).
Tale sensore è un Monolithic Active Pixel Sensor (MAPS), ovverosia il detector e l'elettronica di front-end sono ottenuti da uno stesso blocco monolitico di silicio, questo per soddisfare i requisiti imposti da ALICE, primi fra tutti la bassa densità di potenza ed elevata risoluzione spaziale.
Il lavoro di questa tesi è principalmente costituito dallo sviluppo di un firmware da implementare in un FPGA per il controllo di tale chip e fornire una semplice comunicazione con PC per l'acquisizione di dati. Sono valutate poi le performance del nuovo sistema di lettura e controllo.

1 Monolithic Active Pixel Sensors (MAPS)

1.1 Sensori al silicio

Una particella carica che attraversa un reticolo cristallino di silicio cede energia al reticolo ionizzando i vari atomi, il numero di elettroni così prodotti sarà proporzionale all'energia di tale particella, per tanto inserendo degli elettrodi opportunamente polarizzati e misurando la quantità di elettroni raccolti è possibile risalire all'energia della particella. Tuttavia dato il piccolo valore del band gap del silicio ($\sim 1.12eV$) alla temperatura di $300K$ saranno presenti molti elettroni in banda di conduzione e quindi si misurerà una corrente di buio che renderà molto difficile la rivelazione di particelle cariche.
Per aggirare tale problema si producono due zone di silicio diversamente drogate, una con atomi trivalenti creando così delle lacune nel reticolo (p-type) e una con atomi pentavalenti ottenenedo degli elettroni liberi (n-type), mettendo assieme le due regioni si ottiene la cosiddetta giunzione p-n. Senza applicare alcuna tensione l'eccesso di elettroni nella regione n per diffusione passerà alla regione con eccesso di lacune (viceversa per le lacune nella regione p), ricombinandosi con esse, questo creerà nella zona intermedia una regione di carica spaziale (positiva nel n-type e negativa nel p-type).

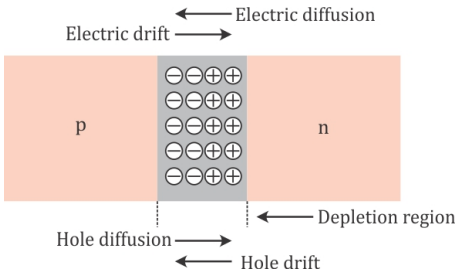

Figura 1.1: p-n junction structure

Questa regione di carica spaziale, chiamata zona svuotata, creerà un campo elettrico e quindi un potenziale che impedirà un' ulterirore diffusione di elettroni (e lacune) questo è chiamato potenziale di built-in (1.1) [1], una particella carica attraversando questa zona crea delle coppie elettrone-lacuna(e-h), che per effetto del campo elettrico saranno raccolte agli elettrodi (ai capi della giunzione). Lo spessore della zona svuotata dipende dal livello di drogaggio delle due regioni e dalla tensione applicata alla giunzione (1.2) [1], solitamente si applica una polarizzazione inversa in maniera tale da aumentare il volume sensibile del detector. I parametri fondamentali per questo tipo di detector e che incidono sulle sue prestazioni (come ad esempio la frequenza con cui riescono a rivelare particelle, efficienza e potenza richiesta per il funzionamento) sono il processo produttivo dei wafer di silicio e la capacità di giunzione C_j (1.3) [1].
Da non trascurare è la corrente di leakage dovuta alla struttura della giunzione , infatti applicando una tensione di bias inversa si rende nulla la corrente di diffusione, ma non si elimina la corrente di drift; nei diodi questa corrente è chiamata appunto corrente di saturazione (o corrente di scala poiché è legata all'area del diodo)((1.4)[1] e (1.5)[2]) [1].

$$V_0 = V_T \ln \frac{N_A N_D}{n_i^2} \tag{1.1}$$

$$W = \sqrt{\frac{2\epsilon_s}{e}\left(\frac{1}{N_A} + \frac{1}{N_D}\right)(V_0 + V_R)} \tag{1.2}$$

$$C_j = A\sqrt{\frac{e\epsilon_s}{2}\frac{N_A N_D}{N_A + N_D}\frac{1}{V_0 + V_R}} \tag{1.3}$$

[1] La prima componente corrisponde alla corrente di diffusione (dipendente dalla temperatura e tensione applicata), mentre la seconda alla corrente di drift

[2] D_p è la diffusività delle lacune (o elettroni) mentre L_p è la loro lunghezza di diffusione, ovvero la lunghezza in cui la densità delle lacune si è ridotta di un fattore 1/e per ricombinazione

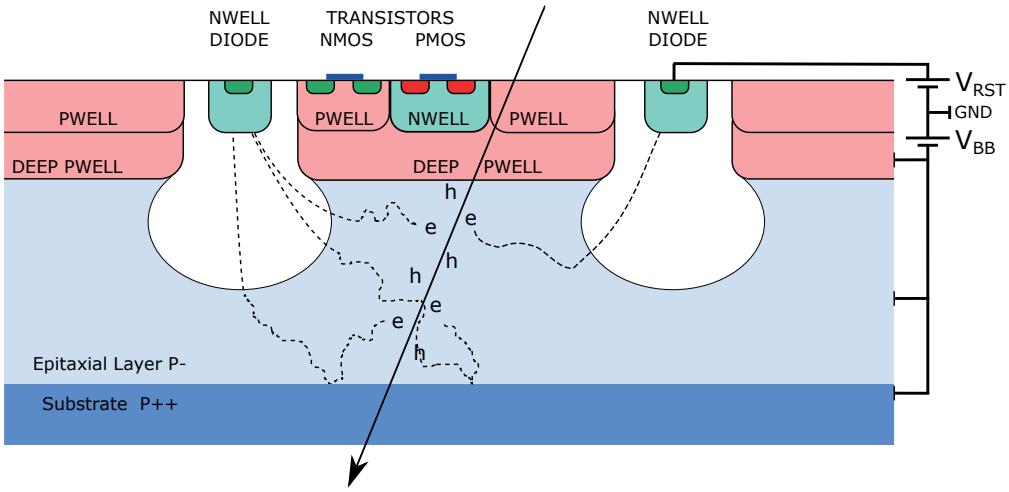

Figura 1.2: Simplified MAPS structure

$$I_{diode} = I_S(e^{V/V_T} - 1) \tag{1.4}$$

$$I_S = Aen_i^2 \left(\frac{D_p}{L_p N_D} + \frac{D_n}{L_n N_A} \right) \tag{1.5}$$

1.2 Struttura di un MAPS

I Monolithic Active Pixel Sensor o MAPS, introdotti negli anni '90, hanno la peculiarità di integrare il diodo collettore e il front-end in un unico wafer di silicio, questo porta a svariati vantaggi quali:

- fanno uso della tecnologia CMOS (continui sviluppi)
- essendo monolitici, l' accoppiamento con i rispettivi amplificatori è semplice (riduzione dei disturbi dovuti alla linee di trasmissione)
- grazie agli attuali sviluppi nei processi produttivi di transistor si possono costruire di dimensione molto ridotta, aumentando così la risoluzione spaziale
- basso consumo in potenza, quindi bassa densità di potenza del singolo chip
- possono essere prodotti con elevata resistenza alle radiazioni (necessario in ambito scientifico)
- elevata parallelizzazione, quindi readout e conversione analog-to-digital molto veloci

La struttura di un MAPS è composta principalmente da tre stati, il primo strato fortemente drogato (p++), chiamato substrato, che funge da supporto per i pixels e per il front-end analogico, un sottile strato poco drogato (p-) chiamto strato epitassiale (qui si ha il volume sensibile del detector) ed infine delle strutture p o n chiamate well le quali ospitano il diodo sensibile (n-well), i pMOS e gli nMOS per gli amplificatori (deep p-well)[3]. Come accennato in precedenza se viene applicata una tensione opposta al verso di conduzione del diodo si aumenterà il volume sensibile (nel chip ALPIDE questa tensione è di 1.8V [2]), se fosse necessario aumentare ulteriormente il volume sensibile nello strato epitassiale è possibile portare a tensioni negative il substrato di supporto (V_{BB} in figura 1.2).

Una particella carica può creare una coppia e-h nei tre strati, se viene creata nelle pwells o nel substrato gli elettroni passeranno nello strato epitassiale per diffusione (favoriti nel passaggio attraverso le giunzioni p-p++ e p-pwell) e una volta entrate nel volume sensibile diffonderanno fino ad entrare nella zona svuotata e verranno raccolte, se invece la coppia viene formata direttamente nello strato

[3]Nel caso di ALPIDE il doping dei vari layer è $N_{Ap++} \sim 10^{18} cm^{-3}$, $N_{Ap-} \sim 10^{13} cm^{-3}$, $N_{Awell} \sim 10^{16} cm^{-3}$

epitassiale l'elettrone rimarrà intrappolato (giunzioni fungono da barriera) quindi diffonderà fintantoché non ricombina (poco probabile, dato il tempo di ricombinaznione molto superiore al tempo di raccoglimento) o entra nel campo elettrico della zona svuotata e potrà essere raccolto. Lo spessore del volume sensibile per i MAPS si attesta attorno alle decine di μm contro qualche centinaio di μm per i detector al silicio convenzionali, questo si traduce in una minore quantità di carica liberata e quindi una bassa risoluzione in energia, pertanto l'identificazione di particelle risulta difficile con questo tipo di sensori, ma ciò in cui eccelle è sua la risoluzione spaziale (l'area del singolo pixel è di circa $800\mu m^2$ [2]), infatti vengono utilizzati digitalmente (hit-no-hit mode) e in stacks in maniera tale da poter avere una traccia della particella.

1.3 Formazione del segnale

Il segnale generato dipende dalla carica rilasciata e rilevata dal diodo collettore (Q_{coll}) e dalla sua capacità (C_p), quest'ultima è funzione sia della capacità di giunzione (C_j, predominante) che dalla capacità parassitica del circuito di amplificazione (C_r) [2].
Il segnale totale al diodo sarà del tipo

$$\Delta V = \frac{Q_{coll}}{C_p} \tag{1.6}$$

Ovviamente per massimizzare il segnale raccolto è necessario minimizzare la capacità, per far questo si applica una tensione al substrato (V_{BB} fino a $-8V$ per il chip ALPIDE) e si riduce il più possibile la sua area (anche per limitare la corrente di leakage I_S), inoltre viene ottimizzato il front-end analogico per ridurre la capacità parassitica, tuttavia riducendo l'area del diodo si riduce anche il suo volume sensibile e quindi la carica raccolta, è necessario quindi uno studio di ottimizzazione dei due parametri; per il chip ALPIDE ad esempio sono stati prodotti svariati prototipi (pALPIDE) con parametri diversi (come ad esempio lo spessore dello strato epitassiale), successivamente sono stati testati ed è stato scelto il chip con le migliori prestazioni.

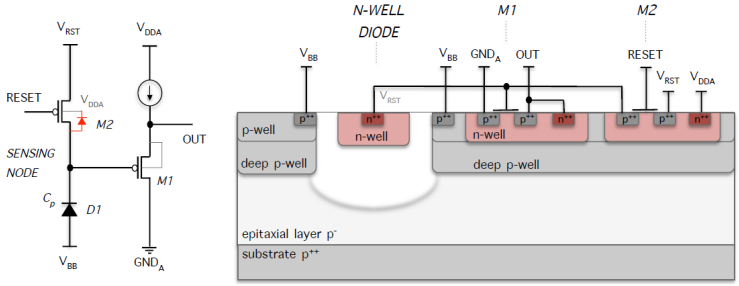

Figura 1.3: Preamplifier stage

Come accennato precedentemente i MAPS sono detector con spessori almeno un ordine di grandezza inferirore rispetto ai sensori a stato solido convezionali, infatti considerando una MIP (Minimum Ionizing Particle) la sua energia ceduta al reticolo è di circa $1.66MeV\ g^{-1}cm^2$, questo per il silicio con densità di $2.3gcm^{-3}$ si traduce in una stopping power di $388eV/\mu m$ [3]. Considerando inoltre che per produrre una coppia e-h sono necessari $3.67eV$[4] [3] si arriva ad una produzione di circa 100 coppie per μm, quindi nello strato epitassiale si avrà una produzione di circa 2500 coppie e-h (una carica di circa 0.4 fC). Per rilevare tale segnale il diodo è portato ad una tensione di reset (V_{RESET} $0.7 \sim 0.8V$ fino ad un massimo di $1.8V$) e una volta che una particella carica genera delle coppie si innesca la scarica della capacità C_p, questo segnale sarà quindi iniettato su un preamplificatore (figura 1.3) poi and un secondo amplificatore per lo shaping del segnale ed infine ad un discriminatore (hit-no-hit mode).

[4] circa tre volte superiore al bandgap del silicio, questo perché per la conservazione della quantità di moto è necessaria l'eccitazione del reticolo, ovvero la produzione di fononi

2 Il chip ALPIDE

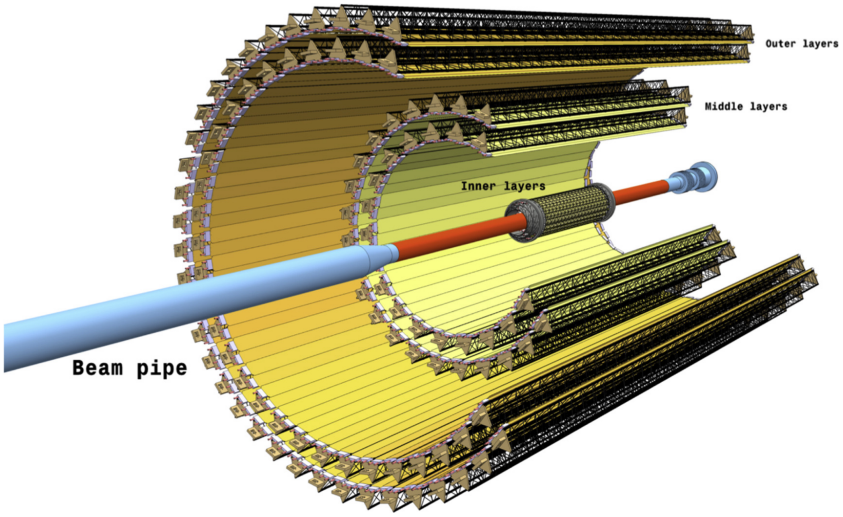

Figura 2.1: ALICE ITS

Il chip ALPIDE deve essere installato come outer barrel module (OB) e inner barrel module (IB) nell'ITS di ALICE, utilizzando lo stesso chip esso deve soddisfare i requisiti più stringenti di entrambi, come ad esempio l'elevata resistenza alla radiazione (per IB) e bassa densità di potenza (per OB) tutto questo tenendo lo spessore sotto i $50\mu m$ [12]; per soddisfare ciò sono stati prodotti diversi prototipi di spessore diverso e poi testati. Si è raggiunto un buon compromesso con un chip avente lo spessore dello strato epitassiale di $25\mu m$ e con alta resistività ($> 1k\Omega \cdot cm$) [12].
Si è utilizzato il processo produttivo TowerJazz 180nm CMOS [2], caratteristica peculiare è la presenza di una deep p-well dove saranno impiantate ulterirori wells (sia p che n) per l'alloggio dei transistor nMOS e pMOS (figura 1.2), il diodo collettore invece è isolato dalla deep pwell questo per essere a contatto con lo strato epitassiale (volume sensibile).

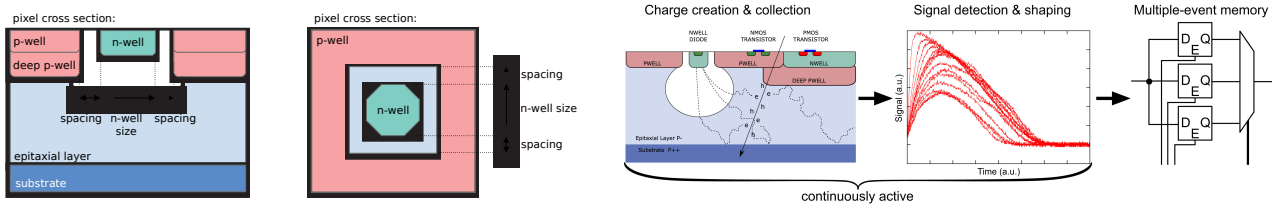

Figura 2.2: Collection diode geometry (left), signal detection stages (right)

I pixels saranno organizzati in una matrice di 512 doppie colonne da 512 righe ciascuna arrivando ad avere una risoluzione di 1024x512 pixels; per ogni doppia colonna la comunicazione tra pixel ed elettronica di readout avviene attraverso un priority encoder che fornisce un indirizzo ad ogni pixel (secondo figura 2.4). Tutto il chip è stato progettato per mantenere bassa la densità di potenza ($< 40mW/cm^2$).
Ad ogni pixel (diodo D1) sarà associato il suo front-end analogico (figura 2.3), si osserva il diodo D0 il quale resetta continuamente il nodo di input pix_in al potenziale (V_{RESETD}), in seguito è presente

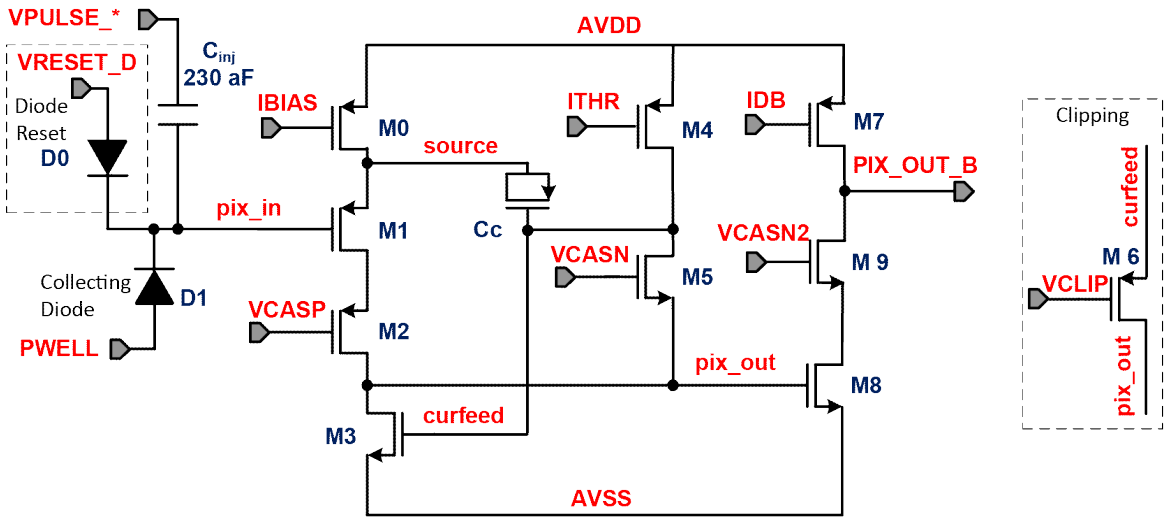

Figura 2.3: Pixel analog front end

il preamplificatore costituito da un cascode[1] (M1 ed M2) con lo scopo di aumentare il SNR (Signal to Noise Ratio) e dare una forma specifica al segnale, il suo output sarà iniettato ad un discriminatore (M4,M5 e clipper M6) e successivamente ad un secondo cascode (M8 ed M9); tutte le tensioni e correnti di bias possono essere settate da opportuni DAC a 8 bit presenti nel chip, queste tensioni e correnti modificheranno il guadagno e la soglia del pixel. Il segnale in uscita PIX_OUT_B attivo basso viene caricato su di un buffer (Multi Event Buffer, MEB) composto da tre latch SR; l'evento viene effettivamente caricato solamente se il segnale di STROBE[2] è alto, in seguito sarà compito del priority encoder leggere lo stato del pixel e propagarlo alla memoria del detector. Caratteristica importante è la presenza di una capacità C_{inj} che ha lo scopo di iniettare una carica di prova in un dato pixel (2.1) [4].

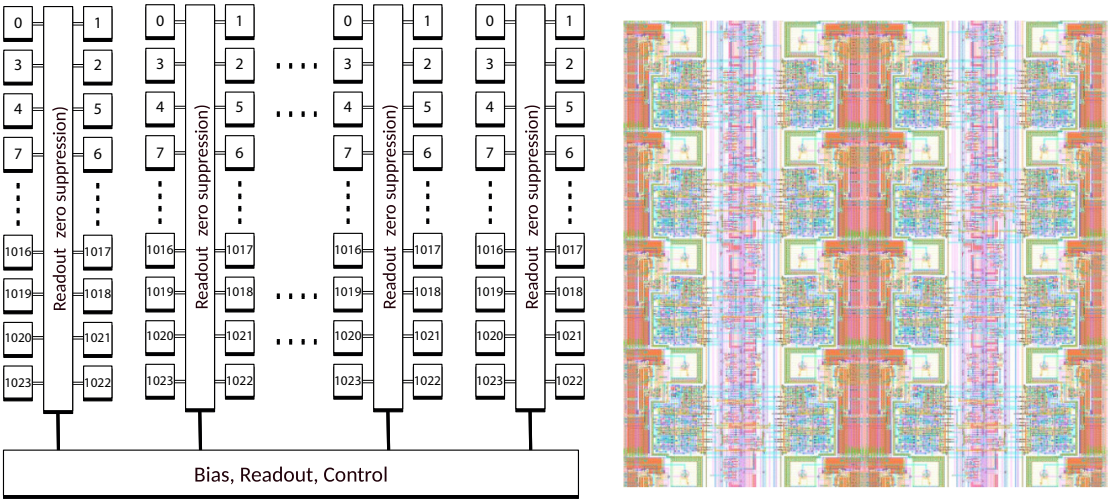

Figura 2.4: Priority encoder addressing (left), ALPIDE diodes and double columns topology (right)

$$Q_{inj} = \Delta V_{pulse} C_{inj} \tag{2.1}$$

Il segnale una volta uscito dal discriminatore (ora è un segnale digitale, PIX OUT B in figura 2.6) se il segnale di strobe è alto viene immagazzinato in uno dei buffer e attraverso un selettore (MEMSEL B) viene selezionato da quale registro verrà preso lo stato del pixel. Sono presenti altre importanti funzioni quali: pulsing, masking, il flushing e ovviamente il reset del MEB.
Il pulsing è necessario per il test di un signolo (o tutti) pixel, questo può avvenire in maniera analogica

[1] si comporta come una configurazione source comune, ma con prestazioni superiori
[2] possibile settare il tempo di STROBE da specifici registri

(APULSE) o digitale (DPULSE). Il pulse test viene attivato con uno specifico registro abilitando il segnale PULSE EN, se è selezioneto il pulsing digitale e si ha un segnale DPULSE viene caricato un '1' su di un registro di memoria indipendentemente dal segnale del dato pixel (si noti la porta OR a monte del MEB), se invece è selezionato il pulsing analogico e si ha un segnale APULSE che comporta la variazione dellala tensione sulla capacità C_{inj} da VPULSEH a VPULSEL iniettando della carica nel preamplificatore descritto precedentemente.
Se un pixel risulta essere rumoroso, è possibile bloccare a '0' il suo segnale, dallo stesso registro per abilitare il pulsing si abilita anche il masking [3] che attraverso una porta AND blocca a '0' lo stato del pixel indipendentemente dal livello dei registri nel MEB o del pixel stesso.
Il flushing resetta semplicemente il valore di uno specifico registro del MEB (utilizzato se tutti i buffer sono pieni ed è necessario immagazzinare un nuovo stato), mentre il reset li resetta tutti, per l'appunto.

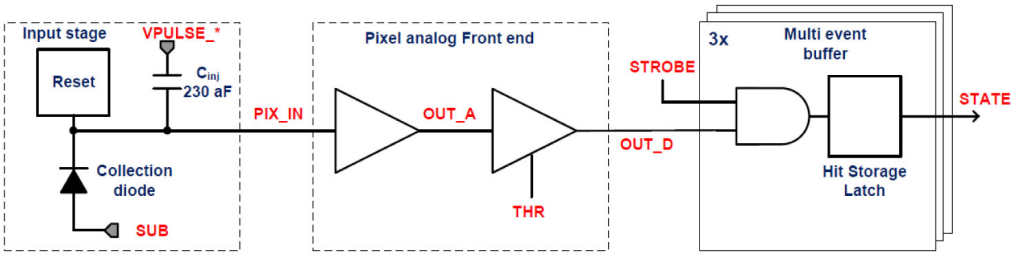

Figura 2.5: Pixel block diagram

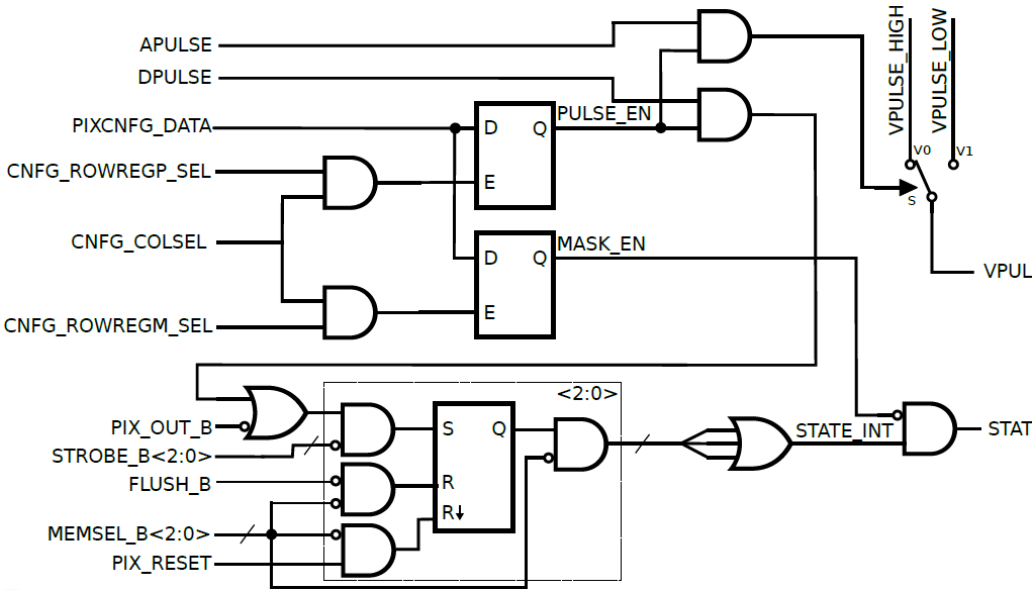

Figura 2.6: Pixel logic

2.1 Interfacce di comunicazione

Dal diagramma a blocchi (figura 2.7) si osservano due linee di comunicazione:

- interfaccia di controllo o CMU (Control Managment Unit) a 40Mbps
- interfaccia di lettura dati, DMU(Data Managment Unit) e DTU(Data Transmission Unit) fino a 1200Mbps

[3] ogni registro ha 16 bit a disposizione

L'interfaccia di controllo è necessaria per fornire le principali comunicazioni fra chip e realtà esterna, infatti da questa interfaccia si possono leggere e scrivere i registri di settaggio e controllo, inoltre da essa è possibile inviare dei comandi di broadcast. Quest' interfaccia ha due porte, una single-ended (CTRL) e una differenziale (DCTRL, con standard LVDS[4]) per offrire un'efficace reiezione dei disturbi.
L'interfaccia di lettura, invece è adibita alla sola lettura della memoria FIFO[5] del sensore. Sono presenti due tipi di comunicazione, una parallela con velocità di trasmissione di 320Mbps e una seriale con velocità selezionablie di 400, 600 o 1200Mbps.

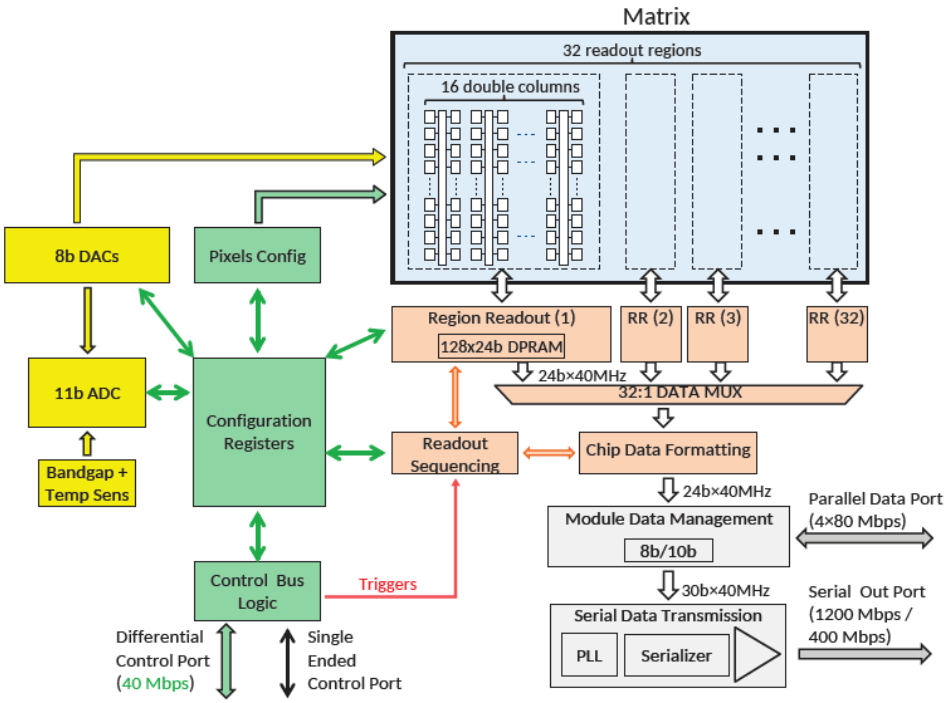

Figura 2.7: Simplified ALPIDE block diagram

2.1.1 Interfaccia di controllo CMU

L' interfaccia di controllo offre due porte di cominicazione una single ended (CTRL) ed una differenziale (DCTRL), quest'ultima offre la possibilità di attivare o meno la codifica Manchester con la convenzione IEEE8 802.3 per indicare la fase/antifase del segnale[6]. La comunicazione è bidirezionale ed half-duplex, ciò significa che non è possibile ricevere e trasmettere dati contemporaneamente.

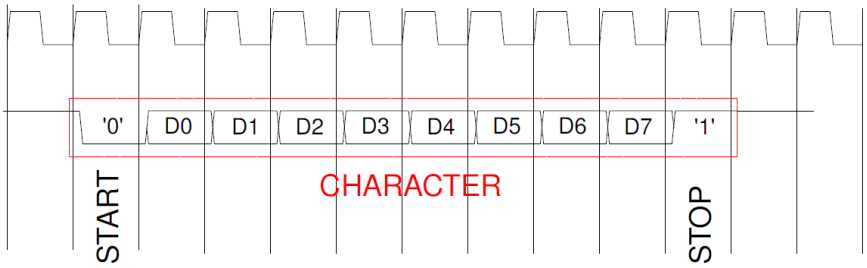

Figura 2.8: Single character, without Manchester encoding

[4]Low Voltage Differential Signaling
[5]First In First Out
[6]ovvero l'invio in entrambi i fronti del clock, nel fronte di salita viene inviato il segnale negato e nel fronte di discesa il segnale non negato

Una transazione corrisponde ad una sequenza di 10 bits, ovvero un bit '0' di inizio, successivamente il byte vero e proprio con la convenzione Less Significant Bit first e un bit '1' di stop. Se ci si trova in stato di idle verrà inviato un '1' logico (nel caso in cui il chip stia ricevendo sarà l'elettronica di acquisizione ad inviare un segnale alto), ciò significa che la massima velocità di trasmissione si riduce a $32Mbps$ (80% di 40Mbps).

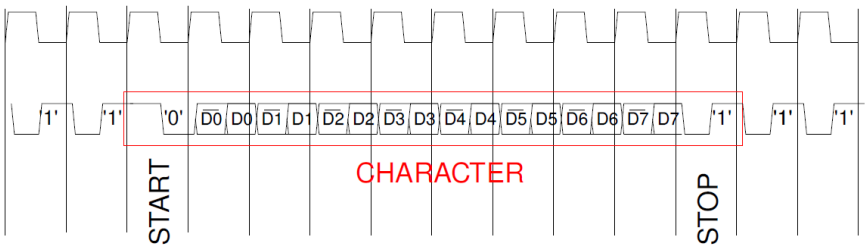

Figura 2.9: Single character, with Manchester encoding (only on DCTRL)

Il gap tra due byte inviati corrispondenti ad una stessa operazione può variare tra 0 e 42 cicli di clock. Le transazioni possibili sono 5:

- comando di BROADCAST: composto da un singolo carattere, è un comando cui reagiscono tutti i chip (e.g reset o trigger, tabella 2.1)
- TRIGGER: è un comando di broadcast, caratteristica specifica di questa operazione è la necessaria velocità di decodifica da parte del chip per evitare ulterirori delay da un segnale di trigger esterno (come ad esempio uno scintillatore), infatti esso reagisce ai 2 less significant bits[7] (per questo la ridondanza di quattro diversi comandi di trigger)
- UNICAST WRITE: composto da 6 caratteri; una word di 16 bits sarà scritta in un registro interno di uno specifico chip. Comincia con il WRITE OPCODE, seguito dal CHIP ID (identificazione del chip), poi due caratteri relativi all'indirizzo del registro scelto ed infine due caratteri corrispondenti al dato da scrivere
- MULTICAST WRITE: simile al UNICAST WRITE con l'unica differenza del CHIP ID, in questo caso sarà un ID di multicast, ovvero il dato verrà scritto sul registro scelto di ogni chip.
- UNICAST READ: composto da 4 caratteri inviati e 3 caratteri ricevuti; può essere solamente unicast. Comincia con il READ OPCODE, seguito dal CHIP ID ed infine i due caratteri dell'indirizzo del registro da leggere, successivamente si ha la fase di *busturnaround* dove il controllo della linea passa dall'elettronica di acquisizione al chip ALPIDE, in seguito il chip invia il suo CHIP ID e i due caratteri corrispondenti al dato immagazzinato nel registro scelto infine si avrà un ulterirore *busturnaround* in maniera tale da tornare al punto iniziale

Una volta inviato lo stop bit dell'ultimo carattere dell' indirizzo da leggere (operazione di READ) si hanno 5 cicli di clock gestiti dal master[8], poi si avranno 5 cicli in cui avviene il passaggio allo slave e infine 5 cicli gestiti dallo slave, nel secondo *busturnaround* si avrà lo stesso procedimento svolto nel verso opposto. In questa fase il master deve rilasciare il controllo del bus per un numero predefinito di cicli di clock(50).

[7] tutti i comandi di trigger terminano con '01'
[8] come master si intende l'elettronica di acquisizione/controllo, mentre come slave si intende il chip

OP code	Hex value	Purpose
TRIGGER	0xB1	Trigger command
TRIGGER	0x55	Trigger command
TRIGGER	0xC9	Trigger command
TRIGGER	0x2D	Trigger command
GRST	0xD2	Chip global reset
PRST	0xE4	Pixel matrix reset
PULSE	0x78	Pixel matrix pulse signal
BCRST	0x36	Bunch counter reset
DEBUG	0xAA	Sample state in shadow register
RORST	0x63	Read out reset
WROP	0x9C	Start UNI/MULTICAST write
RDOP	0x4E	Start UNICAST read

Tabella 2.1: OP codes

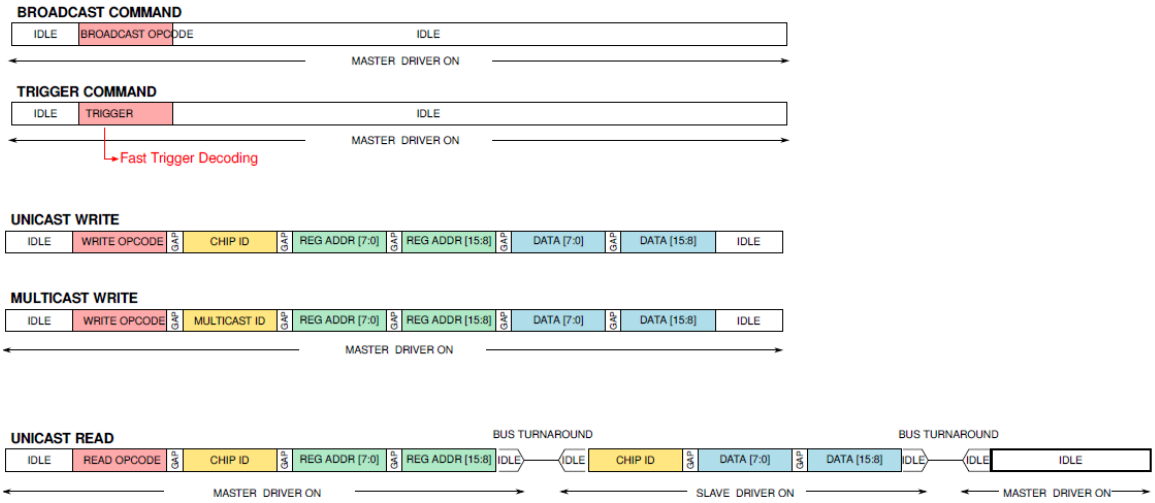

Figura 2.10: Control interface, data transaction

2.1.2 Interfaccia di lettura DMU e DTU

L' interfaccia di lettura dati è caratterizzata da una porta parallela (Data[0:7]) da 8 bits e una porta seriale differenziale (HSDATA P e HSDATA N).
La porta parallela utilizza il clock fornito al chip[9] (40 MHz nominali) ed ha due diversi modalità di invio, nella modalità di default ad ogni fronte di salita del clock l'intero byte di informazione è trasferito attraverso le 8 linee, mentre nella modalità DDR (Double Data Rate) verranno utilizzate 4 linee, ma il byte verrà inviato in due blocchi, uno nel fronte di salita e l'altro nel fronte di discesa (LSB first), ciò comporta una massima velocità di trasmissione di 320Mbps (8b×40MHz o 4b×80MHz)
La porta seriale, come già accennato, può funzionare in 3 diverse velocità di trasmissione (400, 600 e 1200Mbps), sfruttando o meno il DDR. Per questa linea è abilitata di default la codifica 8b/10b che riusulta utilie per una veloce verifica del dato letto, avremo quindi una massima velocità raggiungibile di 960Mbps. Caratteristica importante di questa porte risulta essere la possibilità dell'invio di dati di test secondo una sequenza PRBS-7[10], molto simile alla codifica 8b/10b, così da poter testare le scheda di readout.
I dati relativi ai segnali dei pixel sono immagazzinati in una FIFO con profondità 64 e larghezza 24 bits. I valori relativi a quali pixel hanno reagito ad un evento sono inviati (sia verso la CMU, se

[9] fornito esternamente dal una scheda DAQ o da un ALPIDE chip master, a seconda dell'utilizzo scelto
[10] Pseudo Random Bit Sequencing

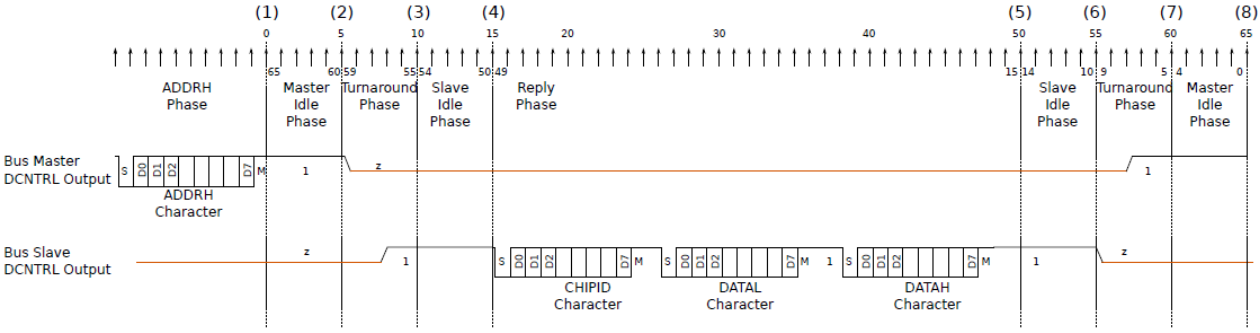

Figura 2.11: Busturnaround on DCTRL port (Manchester encoding disabled)

abilitata, che verso la DMU) attraverso dei pacchetti, ogni pacchetto comincia con un CHIP HEADER e termina con un CHIP TRAILER.

2.2 Trasmissione dati

In tabella 2.2 si vedano tutti i possibili dati inviabili.

Data word	Lenght [bits]	Value [binary]
CHIP HEADER	16	0b 1010<chip id[0:3]><bunch counter[10:3]>
CHIP TRAILER	8	0b 1011<read out flags[3:0]>
CHIP EMPTY	16	0b 1110<chip id[0:3]><bunch counter[10:3]>
REAGION HEADER	8	0b 110<region id[4:0]>
DATA SHORT	16	0b 01<encoder id[3:0]><addr[9:0]>
DATA LONG	24	0b 00<encoder id[3:0]><addr[0:9]>0<hitmap[6:0]>
BUSY ON	8	0b 1111 0001
BUSY OFF	8	0b 1111 0000
IDLE	8	0b 1111 1111

Tabella 2.2: Data word [11]

Di seguito verranno indicate le caratteristiche di ogni data word:

- CHIP HEADER: word di inizio per ogni pacchetto dati, i primi quattro bits sono fissi (necessari per una veloce identificazione del byte), in questa word sono indicati il chip id (locazizzazione dello specifico chip che invierà le posizioni dei pixel colpiti) e il bunch counter (associato ad una sorta di riferimento temporale dall'ultimo trigger)
- CHIP TRAILER: word di fine pacchetto , fornisce eventuali flags di errore nel readout
- CHIP EMPTY: word indicante che nessun pixel ha reagito nel periodo di STROBE, stessa struttura del chip header (eccezion fatta per i primi quattro bits), essa da sola corrisponde ad un pacchetto
- REAGION HEADER: word usata per indicare l' inizio della lettura relativa ad una regione[12] del chip
- DATA SHORT: word contenente la locazione all'interno del chip di un singolo pixel, fornisce il priority encoder di appartenenza (nella data regione) e il suo indirizzo su di esso

[11] i bit mancanti sono impostati a '1'
[12] una delle 32 possibili

- DATA LONG: word usata solamente se il clustering è attivo, fornisce la possibilità di comprimere l'informazione dello stato di al massimo 8 pixels, viene fornita la posizione del primo pixel colpito e se i successivi 7 hanno segnalato un evento
- BUSY ON/OFF: indicano lo stato di busy
- IDLE: word usata come riempitivo se il dato non è pronto per essere trasmesso

Un esempio di pacchetto trasmesso può essere il seguente:
...IDLE, CHIP HEADER (chip interessato e riferimento temporale), REAGION HEADER (regione interessata), un certo numero di DATA SHORT o DATA LONG (identificazione dei pixel colpiti), REGION HEADER , nuovamente DATA SHORT o DATA LONG..., CHIP TRAILER (errori di readout), IDLE...

2.2.1 Read out flags

Le possibile flags fornite alla fine di ogni pacchetto sono ricevute su di un registro a quattro bit (nel CHIP TRAILER) e verranno indicate dal MSB al LSB:

- BUSY VIOLATION: indica che il chip risponde con un pacchetto vuoto (CHIP EMPTY), dovuta alla saturazione della DMU
- FLUSHED INCOMPLETE: indica che il MEB è stato svuotato in maniera tale da aver sempre uno spazio disponibile per un possibile evento, è presente nella sola modalità CONTINUOUS
- STROBE EXTENDED: indica l'estensione della finestra sensibile, dopo l'invio di un ulteriore trigger
- BUSY TRANSITION: indica l'invio di un data BUSY durante il pacchetto in questione

Se si incorre in un errore critico il chip invierà il valore 0xE indipendentemente dalle altre flags asserite [4].

3 Un nuovo sistema di readout e controllo per ALPIDE

In questa tesi è stato sviluppato un sistema di readout e controllo per il singolo chip ALPIDE attraverso una scheda di sviluppo basata su FPGA e la relativa comunicazione con PC.

Il sistema è basato su una un scheda di svipuppo (Digilent Arty A7) con connessione Ethernet, mentre come detector di test è stato utilizzato un chip ALPIDE montato su un PCB (Arduino shield) ad hoc con possibilità di lettura e controllo attraverso microcontrollore Arduino.
Per lettura ed il controllo attraverso Arduino vengono utilizzate delle librerie sviluppate al CERN, queste ultime sono state prese come riferimento e confronto per valutare le performaces del nostro sistema [5]. In questo capitolo illustreremo il sistema a blocchi generale del nuovo sistema elaborato.

Come detector di test viene utilizzato un chip ALPIDE installato sull' Arduino shield. La sola interfaccia di cominicazione disponibile sarà offerta dalla porta single-ended CTRL, mentre il clock principale sarà fornito dalla scheda di sviluppo Arty A7 sui cui si svilupperà il firmware per il controllo del sensore. Tale scheda di sviluppo è equipaggiata con un Artix-7 (Xilinx) la cui velocità massima di trasmissione è 950Mbps utilizzando un protocollo DDR LVDS (Artix-7 con speed grade -1 [6]), ben oltre ai 40Mbps necessari per l'interfaccia di controllo che diventano 80Mbps se si abilita il Manchester encoding sulla lineea differenziale.
Inoltre va fatto notare che scheda la Arty A7 non supporta lo standard LVDS e necessita quindi di un driver presente nell' Arduino shield[1].

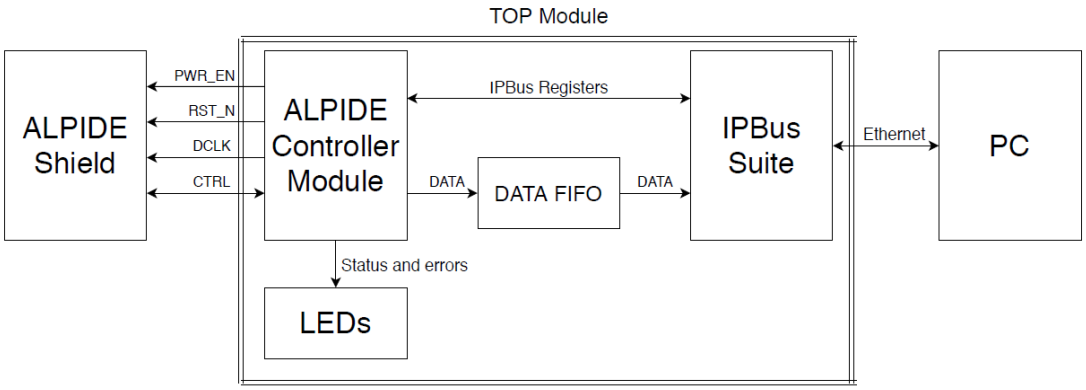

Figura 3.1: Simplified main block diagram

[1]Il sensore ALPIDE accetta solamente segnali di clock differenziali con standard LVDS a 1.8V

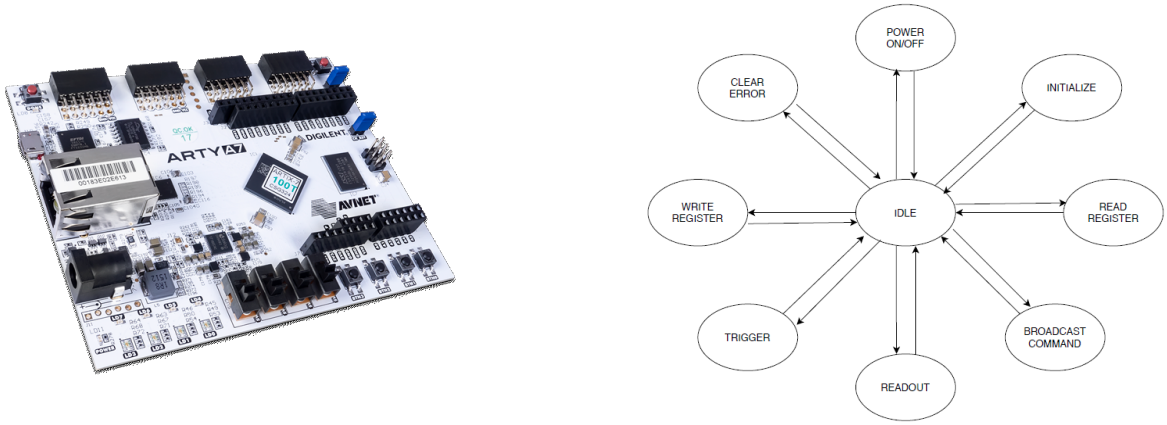

Figura 3.2: Arty A7 board (left), possible routines (right)

3.1 Firmware

Il firmware è stato sviluppato tenendo presente i seguenti vincoli e requisiti:

- deve essere presente una routine di trigger il più veloce possibile
- è necessaria una comunicazione con un PC (via Ethernet o USB)
- la porta CTRL deve essere messa ad alta impedenza nella fase di lettura per evitare che lo slave e il master si contendano la gestione della linea
- prestare attenzione nei CDC (clock domain crossing) tra clock di ALPIDE (40MHz) e clock di IPBus (31.25MHz) per evitare eventuali metastabilità e campionamenti errati dei segnali

Inizialmente sono state modificate le macchine a stati finiti (FSM) sviluppate in un lavoro precedente [10], per la lettura/scrittura di un byte secondo il protocollo illustrato nella sezione 3.1.1; le principali modifiche apportate sono state le seguenti:

- è stato aggiunto un modulo PLL per la generazione del clock (esterno ai processi delle FSMs) con frequenza di 40MHz
- per la FSM di lettura è stato aggiunto un secondo clock sfasato di 90° per il campionamento del segnale, questo per aggirare eventuali ritardi di trasmissione e metastabilità del segnale
- aggiunta una flag di errore nella FSM di lettura

Successivamente è stato sviluppato un modulo con varie routine quali, power off/on, inizializzazione, invio comando di broadcast, trigger, lettura/scrittura di un registro, reset delle flags di lettura e readout continuo.
É importante notare che l'FPGA alla sua accensione o al suo reset si trova nello stato di idle, per eseguire una data routine è necessario impostare il suo valore esadecimale su di un apposito registro di 4 bits (fino a 16 operazioni disponibili) e validare l'operazione con un segnale di start. Tutte le operazioni hanno una durata predefinita, eccezion fatta per il readout continuo per la quale viene fermata con un ulteriore segnale (stop read out). Dallo stato di idle ad ogni fronte di salita del clock viene considerato se c'è stato un fronte di salita del segnale di start[2], se la risposta è affermativa l'FPGA cambia il suo stato su uno relativo alla routine da performare, una volta terminata torna allo stato di idle.
Sono state inserite delle condizioni per l'esecuzione di tali routine:

- l'inizializzazione non può avvenire se prima non si è eseguito il power on

[2]si è ottenuto ciò utilizzando due registri

- tutte le routine necessitano prima dell' inizializzazione, eccezion fatta per il power on/off e il clear degli errori

La comunicazione con il PC si è ottenuta attraverso l' IPBus [11], ovvero un protocollo sviluppato al CERN il quale, sfruttando l'interfaccia Ethernet (o PCIe), modifica dei registri di 32 bit interni al FPGA, è stata quindi implementata tale suite su un modulo (chiamato top module) su cui sono stati inseriti i collegamenti con modulo di controllo di ALPIDE.
Complessivamente sono stati utilizzati dei registri da 16 bit per la trasmissione degli indirizzi dei registri di ALPIDE da leggere/scrivere, un registro per il suo status (ad ogni bit è stato associato un segnale, sola lettura) e un registro per il controllo (in una maniera simile al registro di status, sia lettura che scrittura).

3.2 Software

L' interfaccia con il PC è stata sviluppata attraverso gli strumenti forniti dalla suite IPBus, ovvero un'API chiamata μhal che attraverso il protocollo Ethernet invia e riceve delle transazioni per la modifica di registri interni all'FPGA da 32 byte ciascuno.
Il software sviluppato sarà composto da quattro file:

- vari scripts sviluppati in Python per la decodifica delle hitmap e controllo di ALPIDE
- un file .xml per fornire allo script i dati dei vari slave (registri di status e controllo, memorie RAM di lettura) da comandare, come ad esempio gli indirizzi delle DPRAM (Dual Port RAM) sui cui leggere/scrivere e i loro identificativi
- un ulteriore file .xml per l' identificazione dello slave principale, in questo caso sarà fornito il nome (Arty7) e il suo indirizzo ip (10.10.10.100)
- uno script per la codifica e generazione di un file HDL che mette in relazione gli slave tra FPGA e suite IPBus

Sfruttando tali registri è possibile inviare al PC svariate informazioni come ad esempio status del sistema o eventuali errori rilevati, nonché i dati letti dal chip. Questa API è basata su C++ e Python[3], ci concentreremo maggiormente su quest'ultimo linguaggio data la sua semplicità e flessibilità.
Lo script sviluppato offre una console di comando con svariate funzioni tra cui la lettura e scrittura dei registri del chip ALPIDE i quali sono fondamentali per il controllo dello stesso, una routine di inizializzazione che resetta il chip e imposta i valori su alcuni specifici registri di controllo, ovviamente l'accensione del detector e una funzione di readout continuo (tramite la CMU).
Verrà inoltre introdotto il test del sensore tramite PULSE (analogico o digitale) con l'opzione di impulsare uno o più pixel (fornendo le coordinate degli stessi).
La lettura attraverso la CMU presenta alcune mancanze, come l'incapacità svolgere l'operazione di readout e di invio comandi contemporaneamente, ciò comporta la scarsa reattività del sistema a segnali di trigger inviati dal software.
Nello script saranno fornite varie funzioni (eg lettura/scrittura registro) per il controllo e test del chip, ciò comporta una notevole capacità di modifica per venire incontro alle necessità dei progetti futuri.

[3]Python è piuttosto lento, tuttavia tutte le operazioni che richiedono la massima velocità saranno gestite da firmware (e.g. trigger e read out)

4 Blocchi principali del sistema di readout, controllo e debug

In questo capitolo vengono descritti in dettaglio alcuni blocchi fondamentali del sistema.

4.1 Inizializzazione del chip

Il blocco di inizializzazione consiste nell'invio di alcuni comandi di broadcast e la scrittura (MULTICAST) di alcuni registri di configurazione. Tale routine è stata inserita per minimizzare il tempo di inizializzazione del chip.
Inizialmente si pone a zero logico il segnale di reset per un ciclo di clock in seguito lo si riporta ad 1 (il reset è attivo basso nel caso di ALPIDE), in seguito vengono mandati due comandi di broadcast ovvero il global reset e il pixel matrix reset, questo per eliminare qualunque tipo di settaggio precedentemente impostato, successivamente vengono scritti i registri per il settaggio del chip quali:

- impostazioni della CMU e DMU: disabilitazione del Manchester encoding e del DDR
- settaggio del DAC relativo al VRESETD: impostato a 1400mV
- reset di eventuali configurazioni della matrice dei pixel (pulsing o masking)
- impostazioni del Mode Control Reg: read out triggered, clustering non abilitato, frequenza di lettura del pixel a 20MHz e abilitazione di read out tramite CMU

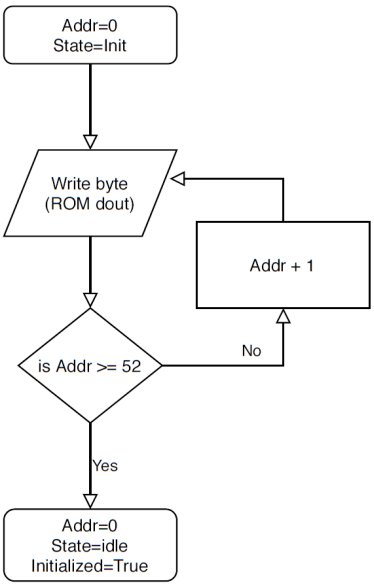

Figura 4.1: Initialization

Infine viene mandato un comando di reset del read out[1].
Per l'invio di questi byte[2] è stata introdotta una ROM tramite la primitiva XPM_MEMORY_SPROM [7] a singola porta con depth di 256 locazioni e width di 8 bit.
La routine consiste in una macchina a stati che inizia da un indirizzo base (impostato a 0x00) e ad ogni byte inviato incrementa l'indirizzo di un' unità, una volta raggiunto il cinquantaduesimo indirizzo la FSM si resetta e fornisce un segnale di inizializzazione completata, quindi l' FPGA torna allo stato di idle. Per costruzione tra due byte inviati è presente un gap di tre cicli di clock, quindi per la completa inizializzazione di ALPIDE ci si aspettano 673 cicli[3], corrispondenti a 16.825μs.

4.2 Rountine di readout

[1] RORST in tabella 2.1
[2] un totale di 52 byte
[3] 10 bit per byte inviato e 51 gap di 3 cicli, quindi 52·10+51·3=673

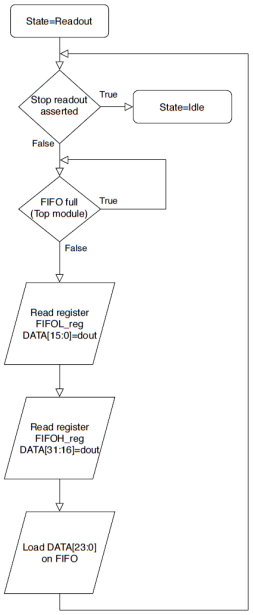

Figura 4.2

La routine segue il protocollo descritto in sezione (2.1.1), quindi verrà inizialmente letto il registro FIFOL (indirizzo 0x0012) e poi il registro FIFOH (indirizzo 0x0013), una volta ottenuto il dato da 32 bit (di cui gli ultimi 8 saranno fissi a '0') verrà segnalata la possibilità di lettura, quindi verrà caricato il dato (ora da 24 bit per minimizare l'utilizzo di risorse nell' FPGA) in una FIFO generata tramite la primitiva XPM_FIFO_SYNC [7]. Tale FIFO avrà width di 24 bit e depth di 8192, avrà inoltre dei segnali di controllo, quali empty, full e prog_full(darà un 1 logico se la FIFO ha almeno 512 dati al suo interno). Questa routine viene ripetuta fintantoché non viene asserito un segnale di "stop read out", una volta asserito la routine termina l'ultima lettura (quindi si ferma solo dopo aver letto il registro alto) e solo successivamente l'FPGA torna in stato di idle.
Il read out tramite CMU è inefficiente, infatti per la lettura di un dato di 24 bit è necessario leggere il contenuto di due registri (FIFOL contenente i primi 16 bit e FIFOH contenente i restanti 8), per questo motivo sono necessari almeno 109 cicli di clock[4] per ogni registro, per tanto avremo un dato ogni 218 cicli che corrispondono ad un throughput teorico di 4.40Mbps[5]. Inoltre, dato che si sta utilizzando la linea di controllo per la lettura, non sarà possibile inviare comandi al chip (per questo la necessità di implementare anche la linea di lettura dedicata).

4.3 Readout continuo tramite PC

Funzione fondamentale del firmware e software sviluppato è il test e controllo del detector ALPIDE, è d'obbligo quindi avere una maniera di collaudare il readout. Come già esposto precedentemente i dati contenuti nella FIFO di ALPIDE sono spostati in ulteriore FIFO, qui una volta ricevuto il segnale di prog_full (almeno 512 bit nella FIFO) comincia la trasmissione di un blocco di 512 dati nella DPRAM (Dual Port RAM) della IPBus suite. Su una porta della memoria saranno presenti le connessioni dell'IPBUS con clock principale impostato a 31.25MHz, mentre la seconda porta sarà collegata al top module del codice sviluppato con clock di ALPIDE ovvero 40MHz (in questa maniera si evitano i cosiddetti CDC o Clock Domain Crossing). Una volta che saranno trasferiti i 512 dati sarà asserita una flag (memory readable), quindi il software leggerà il blocco e inserirà i dati in una lista (RawData) presente nello script Python e successivamente invierà il segnale di avvenuta lettura (memory read)[6].
Un fattore di criticità è la durata del trasferimento del blocco dalla DPRAM alla lista nello script, infatti mentre avviene tale trasferiemnto, il readout è ancora in esecuzione ed è possibile che venga riempita completamente la FIFO del top module, ciò comporta una momentanea interruzione del readout riducendo il throughput massimo. La lettura continua fintantoché non viene terminato dall' utente o si incorre in un errore critico, una volta terminato il software invia un segnale per bloccare l'intero processo di read out (stop read out).
La lista RawData verrà in seguito decodificata secondo il protocollo di trasmissione descritto in sezione 2.2 e verrà quindi generata una mappa del chip

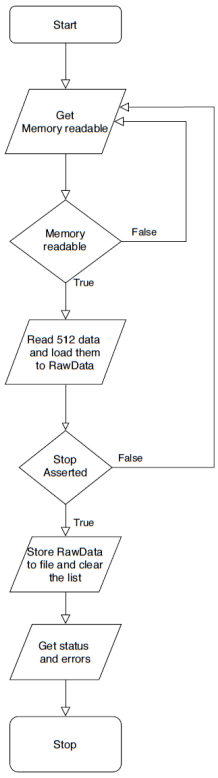

Figura 4.3

[4]sempre considerando 3 cicli di gap tra i byte solo all'invio, quindi 40 per i byte inviati, 9 di gap, 30 per entrambi i bus turnaround e 30 per i byte ricevuti, supponendo l'invio di ALPIDE perfetto

[5]$40\text{MHz}\frac{24bit}{218} = 4.404\text{Mbps}$

[6]La FIFO è stata costruita con depth di 8192 locazioni in maniera tale da permettere il trasferiemnto e lettura di 512 dati senza incorrere in problemi di FIFO full

che identificherà il numero di eventi rilevati da ogni pixel secondo una scala di colori.

4.4 Emulazione di ALPIDE con FPGA

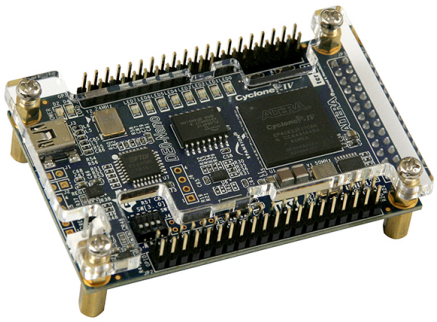

Figura 4.4: DE-0 nano board

Per testare il firmware sviluppato, è stata inizialmente impiegata una seconda scheda di sviluppo (DE-0 nano equipaggiata con un CYCLONE IV di ALTERA [8]) utilizzata come emulatore del chip ALPIDE.
Su questa scheda è stato installato un firmware che reagiscee al solo comando di lettura (0x4E, unicast read), una volta identificato tale comando, verranno letti i vari byte mandati quali chip id ed indirizzo del registro da leggere. Successivamente lo slave invia il suo chip id (fissato a 0x12) e come dato presente nel registro scelto invia l'indirizzo inviato dal master oppure (per la verificare il corretto invio dei dati) un numero crescente.

In questa maniera si verificano:

- se il master invia e decodifica correttamente i byte
- la flag per un incorretto chip id ricevuto
- varie flag di errore nella lettura, come ad esempio la mancata ricezione dello stop bit(read error) oppure il superamento del numero massimo di cicli di clock nell'invio dei byte (slave timeout)
- velocità di trasmissione con clock a 40MHz

Nel firmware della scheda di emulazione si è implementato inoltre un metodo per verificare l'invio di dati nella routine di readout. In particolare è stato introdotto un semplice counter (tale conter verrà incementato solamente alla lettura di entrambi i registri 0x0012 e 0x0013, corrispondenti a FIFOL e FIFOH [4]), così facendo saranno inviati dei numeri crescenti, nel caso in cui venga inviato un indirizzo diverso dai due sopracitati il chip fornirà il valore 0xFFFF; la verifica è fatta tramite software, se un dato non rispetta la sequenza verrà incrementata una variabile ed una volta interrotto tale test il software fornirà il numero di dati corrotti, il totale di dati inviati e il throughput effettivo; i risultati di tale test saranno forniti nella sezione dedicata alle prestazioni del sistema (sezione 5.1 e 5.2).

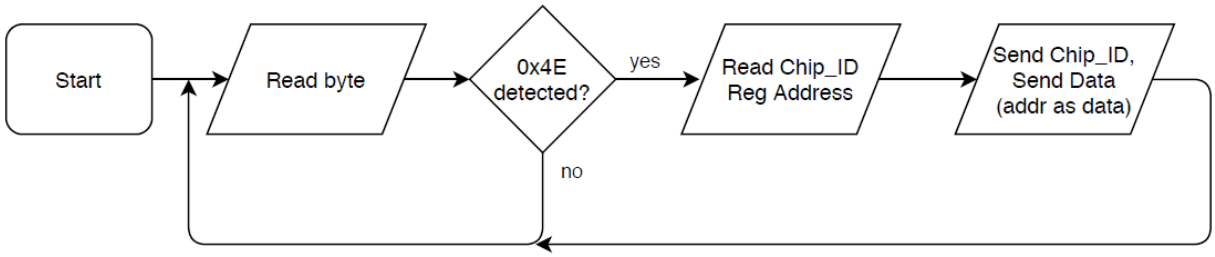

Figura 4.5: ALPIDE emulator block diagram

5 Prestazioni, test e misure

In questo capitolo vengono descritte le performances del sistema sviluppato, come le velocità di trasmissione ottenute, il rate di invio di pacchetti errati e i miglioramenti ottenuti rispetto ad un sistema gestito da Arduino.

5.1 Test corruzione dati

Utilizzando il sitema "fake-ALPIDE" (abilitando la modalità counter) si è proceduto ad eseguire un test di corruzione dati inviati.
L'apparato di test è costituito dalle due schede di sviluppo collegate tra loro con dei semplici cavi jumper di lunghezza circa 10cm, il sistema viene posizionato in una scatola schermata e collegata al GND degli FPGA per attenuare eventuali distrubi esterni.
Il test è durato circa 3 giorni ($2.429 \times 10^5 s$) spedendo 4.169×10^{10} dati (1.001×10^{12} bit). Ciò corrisponde ad una velocià di trasmissione di 4.120Mbps[1], durante tale test non si sono osservati dati corrotti.

5.2 Velocità di lettura

Tutti i codici sviluppati hanno preso come riferimento di partenza le librerie scritte per Arduino-ALPIDE, tuttavia si sono migliorate alcune importanti caratteristiche.
Arduino si basa su di un microcontrollore (ATmega328P) con frequeza di clock di 16MHz [9] e la sua programmazione avviene attraverso un linguaggio molto simile al C++, questo comporta una velocità non ottimale per questo tipo di sensore. Infatti il clock misurato all'oscilloscopio si attesta sugli 80 kHz, inoltre data la natura sequenziale del linguaggio sopracitato per l'invio di byte è necessario prima inviare un periodo di clock ed in seguito leggere/scrivere sulla linea di controllo, ciò comporta un bit inviato ogni 15 μs traducendosi in un ritardo di trasmissione ed un throughput effettivo per il singolo byte di 48kbps, mentre per il read out di 7.2kbps[2].
La scheda Arty A7 d'altro canto è basata su di un FPGA. Ciò comporta il notevole vantaggio di poter eseguire più operazioni contemporaneamente (come ad esempio la generazione del clock e il campionamento del segnale). Un ulteriore collo di bottiglia per Arduino risulta essere l'interfaccia di comunicazione Master-PC basata su UART (con baudrate limitata a 9600), mentre l' Arty A7 utilizza un' interfaccia Ethernet con throughput massimo di 100Mbps.

La differenze tra Arduino ed Arty A7 hanno come conseguenza una performance di lettura radicalemente diversa, il primo è limitato a circa 7.2kbps (supposto un clock di 60kHz), mentre per il secondo ci si aspetta un incremento di circa tre ordini di grandezza.
Il singolo byte verrà trasmesso in 10 cicli di clock, ciò corrisponde ad un throughput di 32Mbps (questo perché sono trasmessi i byte di inizio/stop). Il readout continuo, invece risulta avere un throughput inferiore, infatti per leggere un dato composto da 24 bit è necessario leggere due registri e tenendo conto anche dei gap tra i byte inviati[3] saranno necessari almeno 218 cicli di clock, corrispondente a

[1] in questo caso dato che la macchina a stati nel fake-ALPIDE per l'invio di dati è la stessa presente nella scheda Arty A7, saranno introdotti dei gap anche in invio, quindi un numero totale di cicli necessari pari a 230 corrispondente ad un throughput massimo teorico di 4.174Mbps

[2] senza alcun tipo di gap si avranno 200 cicli per la lettura di un dato da 24 bit

[3] si suppone l'invio di byte di ALPIDE perfetto, quindi senza gap

una velocità massima teorica di 4.404Mbps.
Con una lettura di raggi cosmici durata 17h si è osservato un throughput di 4.30Mbps. Viene fatto notare che se non si gestiscono correttamente i CDC il throughput, misurato nelle stesse condizioni del test sopracitato, viene dimezzato portandosi a 1.86Mbps.

	Maximum throughput [Mbps]	Effective throughput [Mbps]
Fake-ALPIDE	4.174	4.120
ALPIDE	4.404	4.304
ALPIDE (bad CDC managment)	4.404	1.859

Si osserva come una lettura senza decodifica porti ad un throughput di 98.7% del valore massimo raggiungibile, mentre inserendo la decodifica si raggiunge il 97.7%. La leggera differenza con il throughput massimo è dovuta principalmente all'invio delle flag della dual port ram ("mem readable" e "mem read").
Si osserva che per raggiungere una velocità di trasmissione superiore è possibile utilizzare la linea veloce (seriale o parallela), così facendo si aumenterebbe di almeno un paio di ordini di grandezza il suo sample rate[4], tuttavia per fare questo sarebbe stato necessario un lavoro più ampio di quello concesso dai tempi imposti per le tesi triennali.

5.3 Test soglia

Viene ora svolto un test per la stima della carica di soglia al quale il singolo pixel risponde. Inizialmente è stato iniettato un segnale di PULSE su tutti i pixel con una carica iniettata di circa[5] 2000 e^-, come si può osservare della figura 5.1 non tutti i pixel rispondono a tale stimolo, questo è dovuto al fatto che viene sovraccaricata l'elettronica di gestione dell'impulso; si è optato quindi ad impilsarei solamente 32768 pixels per volta (1024 pixels per regione).Vengono stimolati con un impulso di durata 650ns, un delay tra PULSE e STROBE di 475ns e una durata di STROBE di 75ns.
Per ogni carica iniettata (da 0 e^- a 1600 e^- a passi di 10 e^-) vengono svolte 50 misure ricavando così la frazione $\epsilon_i = \frac{N^i_{fired}}{N^i_{inj}}$ per ogni pixel, l'errore verrà valutato come binomiale $\left(\sigma_{\epsilon_i} = \sqrt{\frac{(\epsilon_i)(1-\epsilon_i)}{N^i_{inj}}}\right)$.
Assumendo che il rumeore elettronico segua un andamento gaussiano, fittiamo[6] la curva ottenuta in funziuone della carica iniettata con la funzione (5.1) dove μ(Threshold) e σ(Noise) saranno i parametri di tale gaussiana.

$$\frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right] \tag{5.1}$$

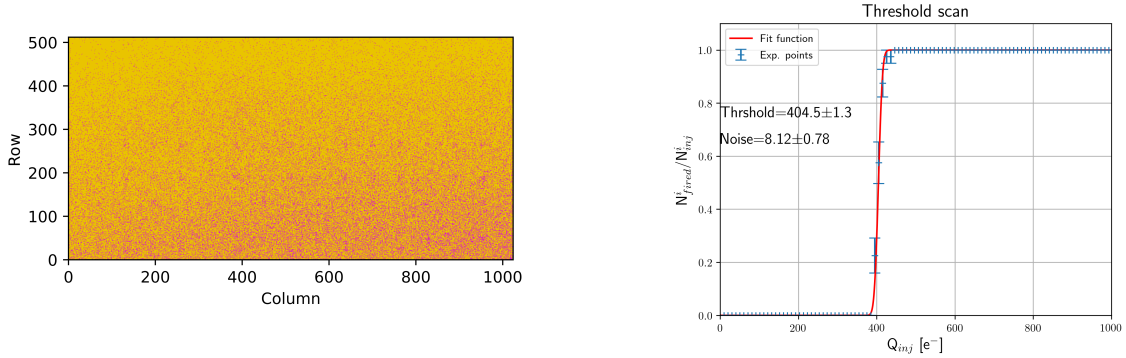

Figura 5.1: PULSE test (left), threshold scan (right)

[4] per ITS è necessario un sample-rate di 100kHz per collisioni Pb-Pb
[5] $\frac{(1.8-0.37)\cdot 230\cdot 10^{-18}}{1.602\cdot 10^{-19}}$ ovvero con VPULSEH e VPULSEL rispettivamente a 1.8V (massimo) e 0.37V (minimo), questo considerando la capacità di 230aF [4]
[6] usati gli algoritmi forniti dalla libreria python SciPy

Una volta ottenute tutte le soglie ed i relativi rumori elettronici li inseriamo in un istogramma e forniamo le relative medie e deviazioni standard. Facciamo notare che non sono stati rilevati tutti i pixel, questo perché sono stati esclusi quelli rumorosi (7) e quelli difettosi (2).

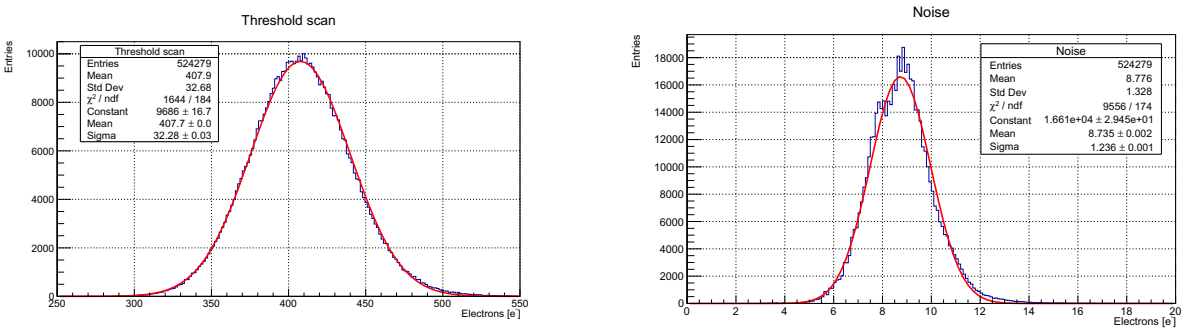

Figura 5.2: Threshold scan (left), Noise scan (right)

Threshold[e^-]	Noise[e^-]
408 ± 33	8.8 ± 1.3

Si nota come la distribuzione delle soglie discosti leggermente dal fit gaussiano, si ha una coda leggermente maggiore a soglie più elevate; la distribuzione dei rumori presenta un andamento simile, tuttavia il fit risulta essere meno pulito attorno alla media.

5.4 Studio di risposta del chip illuminato da diverse particelle ionizzanti

Vengono ora analizzate le dimensioni (numero di pixel) dei cluster con varie particelle incidenti quali alfa (da sorgente di Am-241), beta (da sorgente Sr-90), raggi X da 64keV e raggi cosmici (μ). Questi ultimi verranno raccolti in una finestra temporale più ampia (4 ore) rispetto alle altre particelle di test (30 minuti).
Inizialmente verranno mascherati i pixel rumorosi, verrà poi schermato il detector dalla luce con un panno nero. Verranno in seguito eliminati tutti i cluster composti da un singolo pixel, poiché considerati come rumore, inoltre saranno eliminati anche i cluster composti da più di 26 pixel per eliminare eventuali overlap di due o più tracce. Viene quindi considerata la media e come errore statistico viene considerata deviazione standard della distribuzione ottenuta.
La durata dello STROBE sarà settata a 5000 cicli di clock ($125\mu s$), mentre il gap tra due STROBE è impostato a 65000 cicli ($1.625\ ms$), corrispondente ad un sample rate di 571Hz.

Test particle	Clusters detected	Size [pixels]	Area [μm^2]
Alpha	5700/30min	8.2 ± 6.1	$(6.4 \pm 4.8) \times 10^3$
e^-	475/30min	5.1 ± 3.2	$(4.0 \pm 2.5) \times 10^3$
X ray	1343/30min	4.1 ± 1.4	$(3.2 \pm 1.1) \times 10^3$
μ	348/4h	4.8 ± 2.9	$(3.8 \pm 2.3) \times 10^3$

Tabella 5.1: Single pixel area = $783.632\mu m^2$ [4]

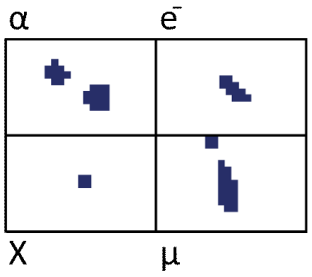

Figura 5.3: Clusters shape

Come ci si aspetta le particelle alfa lasciano una traccia che coinvolge più pixels (e con maggiore dispersione) data la lora densità di ionizzazione specifica, mentre i raggi X ionizzano un'area piàù limitata e dispersione più contanuta; gli elettroni del Sr-90 e i raggi cosmic, invece hanno tracce che coinvolgono meno pixels (sono particelle in pratica al minimo di ionizzazione).

6 Conclusioni

Il lavoro svolto in questa tesi è stato incentrato nello sviluppo di un nuovo sistema per la lettura e controllo di un singolo chip ALPIDE sfruttando le caratteristiche di programmazione molto flessibili degli FPGAs.
Il passaggio da un sistema basato su Arduino a quello basato su FPGA ha comportato notevoli miglioramenti, in termini di flessibilità di programmazione e soprattutto in termini di prestazioni. La transizione da un linguaggio sequenziale (C++ dell' Arduino) ad uno con più possibilità di parallelizzazione (VHDL dell' Arty A7) comporta molta flessibilità e una riduzione drastica dei tempi morti, infatti è stato possibile eliminare il delay del campionamento del segnale dopo l'invio di un periodo di clock, comportando un aumento di velocità di trasimssione di due ordini di grandezza.
Si nota che, per mantenere la mole di lavoro compatibile con i vincoli di tesi triennale, abbiamo basato il sistema sull'utilizzo della linea di controllo. Questo comporta una forte limitazione, per esempio, al momento non è possibile inviare alcun tipo di segnale di controllo mentre la lettura continua è attiva, inoltre lo stesso processo è inefficiente per costruzione (necessari oltre 200 cicli di clock per un singolo dato da 24 bit).
Per ovviare a tale problema è possible utilizzare l'interfaccia di lettura veloce dedicata (DMU e DTU), riuscendo così a passare dai circa 4Mbps ad un massimo di 1200Mbps e inviare la massiva quantità di dati attraverso una seconda porta ethernet (da 1Gbps o preferibilmente da 2.5Gbps) la quale invia pacchetti raw ethernet verso un PC di decodifica.
Un' altra soluzione valida è l'utilizzo delle memorie RAM presenti nelle schede di sviluppo (ad esempio la scheda Arty dispone di una RAM DDR3L da 256MB con velocità di trasmissione massima di 667Mbps), così facendo è possibile caricare i dati letti in tale memoria ed in un secondo momento scaricarli su PC tramite ethernet o USB senza quindi introdurre limitazioni nel readout.
Rimangono ancora alcune funzionalità da introdurre quali: la lettura della porta differenziale di controllo (con codifica Manchester), inserire un sistema di trigger esterno e il miglioramento del software sviluppato per il controllo di più chip simultaneamente (utilizzati ad esempio nello stesso ITS di ALICE o in generici telescopi al silicio). Questo implica un progetto più ampio che verrà approfondito in futuro.

Bibliografia

[1] Adel S. Sedra, Kenneth C. Smith. *Circuiti per la microelettronica*. 4^a Edizione, EdiSES s.r.l., Napoli.

[2] Miljienko Šuljić. *Study of Monolithic Active Pixel Sensors for the Upgrade of the ALICE Inner Tracking System*. Università degli studi di Trieste. A.A. 2016-2017.

[3] M. Tanabashi et al. (Particle Data Group), *Phys. Rev.* D 98, 030001 (2018)

[4] ALICE ITS ALPIDE development team. *ALPIDE Operations Manual*. Ver. 0.3, July 2016

[5] Magnus Mager, *ALPIDE - Arduino reference library for the ALPIDE shield*. 2017

[6] Xilinx, *Artix-7 FPGAs Data Sheet:DC and AC Switching Characteristics*. DS181 (v1.25) June 18, 2018

[7] Xilinx, *FPGA and Zynq-7000 SoC Libraries Guide*. Vivado Design Suite 7 Series, UG953 (v2018.3) December 5, 2018

[8] Terasic, *DE0-nano User Manual*. `www.terasic.com`

[9] Arduino UNO, `https://store.arduino.cc/arduino-uno-rev3`

[10] Andrea Nicolai, *Lettura veloce con FPGA del sensore a pixel ALPIDE per la realizzazione un tracciatore compatto per particelle cariche*. Università degli studi di Padova. A.A. 2018-2019

[11] IPBus Documentation, `https://ipbus.web.cern.ch/ipbus/`

[12] van Hoorne J. W. *Study and Development of a novel Silicon Pixel Detector for the Upgrade of the ALICE Inner Tracking System*. PhD thesis, Technische Universit at Wien, 2008.

[13] Repository Github con software e firmware, `https://github.com/Gabriele-bot/ALPIDE_carrier`