UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Sviluppo di un rivelatore beta di grande area basato sui chip ALPIDE per il progetto ISOLPHARM

Relatore

Prof. Marcello Lunardon

Correlatore

Dr. Davide Chiappara

Laureando

Giovanni Zago

Anno Accademico 2021/2022

Indice

Introduzione

La collaborazione ISOLPHARM ha come obiettivo la produzione di un'ampia gamma di radionuclidi ad elevata purezza per uso medico, sia in ambito diagnostico che terapeutico, sfruttando delle facilities implementate presso i Laboratori Nazionali di Legnaro dell'Istituto Nazionale di Fisica Nucleare. Nell'ambito di questo progetto è sorta la necessità di realizzare un rivelatore in grado di produrre immagini bidimensionali di colture cellulari, contenute in vetrini o sottili strati di scaffold, a cui è stato somministrato il radiofarmaco, in modo da misurarne la quantità contenuta. Attualmente misure di questo tipo vengono effettuate attaccando dei fluorofori alle molecole del farmaco, tecnica che richiede l'impiego di una grande quantità di farmaco che deve anche essere leggermente alterato in modo da permettere l'aggancio del fluoroforo. Con la tecnica di imaging che verrà trattata in questa tesi, invece, si andrà a misurare direttamente la radioattività emessa dalle cellule che hanno trattenuto il radiofarmaco.

Il lavoro svolto e riportato in questa tesi ha avuto come obiettivo quello di capire come utilizzare i chip ALPIDE per realizzare un rivelatore che soddisfacesse le caratteristiche suddette. Partendo dallo studio delle caratteristiche tecniche di ALPIDE, legate soprattutto al suo funzionamento digitale, è stato realizzato del firmware implementato successivamente su un'FPGA, che ha permesso di comunicare con il chip e processare le misure da esso effettuate. Siccome per motivi logistici non è stato possibile ottenere alcune componenti elettroniche strettamente necessarie per poter utilizzare ALPIDE, il funzionamento del firmware è stato accertato solamente tramite simulazioni e non misure vere e proprie.

Nel Capitolo 1 verranno presentate le principali caratteristiche di ALPIDE, i protocolli e le interfacce adibite al controllo del chip e alla trasmissione dei dati. Nel Capitolo 2 verrà presentato il firmware sviluppato e integrato nell'FPGA. Infine, nel Capitolo 3 verranno riportati i risultati di alcune simulazioni effettuate per verificare il funzionamento del firmware.

Capitolo 1

Il chip ALPIDE

ALPIDE [1] è un chip per la rivelazione di particelle basato su *Monolithic Active Pixels*, ideato e costruito per la realizzazione dell'ITS2 (*Internal Tracking System 2*) dell'esperimento ALICE (figura 1.1), nell'ambito degli aggiornamenti apportati all'LHC per l'avvio del Run 3.

Il chip ALPIDE è fabbricato sfruttando la tecnologia CMOS Imaging Sensor di TowerJazz, ed è caratterizzato da una *minimum feature size* pari a 180 nm.

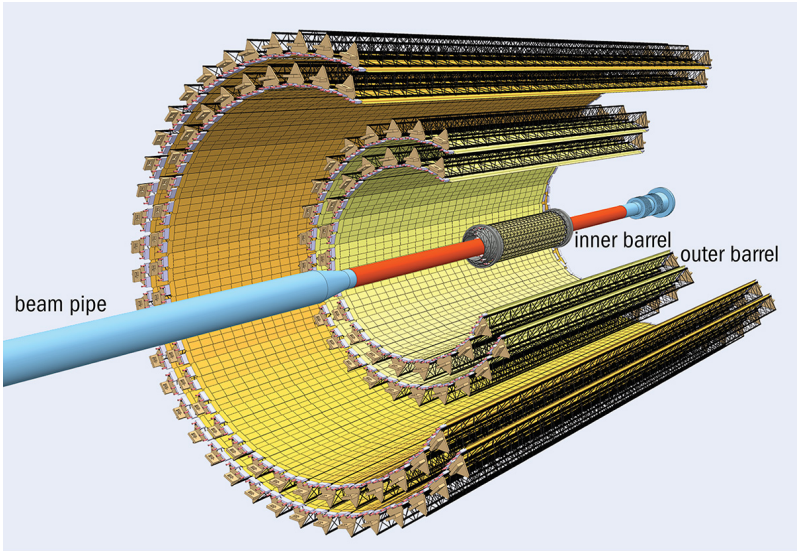

Figura 1.1: Sezione tridimensionale dell'Internal Tracking System 2 dell'esperimento ALICE.

1.1 CCD, Hybrid Chips e MAPS

La caratteristica fondamentale del chip ALPIDE è quella di essere un **MAPS** (*Monolithic Active Pixel Sensor*) [2] . L'origine di tale tecnologia risale all'inizio degli anni '70, quando furono introdotti i primi *pixel sensors*, ovvero dei chip di silicio su cui veniva realizzata una matrice di *charge-collection sites*, che servivano a raccogliere gli elettroni scalzati dal silicio per effetto fotoelettrico, in modo da generare un segnale.

I primi pixel sensors ampiamente utilizzati sono stati i **CCD** (*Charge-Coupled Device*), il cui funzionamento si basa sul fatto che la carica raccolta da ciascun pixel viene trasferita riga per riga, in seguito all'invio al dispositivo di una serie di impulsi di tensione, fino a un circuito di output, costituito da un *readout register* (indicato con R nella figura 1.2 a sinistra) che incanala la carica corrispondente a un'intera riga del chip verso l'*output node*.

Qui la carica va a modulare la tensione presente ai terminali di un transistor, che trasferisce l'informazione al di fuori del chip come segnale analogico. Le limitazioni maggiori dell'utilizzo dei CCD come dispositivi di tracking di particelle sono la scarsa tolleranza alla radiazione e il lungo tempo di readout ($\sim 10^{-2}$ s), dovuto principalmente al fatto che la carica raccolta in ogni pixel, per generare un segnale, deve passare attraverso un unico circuito di output.

Ciò portò allo sviluppo degli **Hybrid Chips**, ovvero dei dispositivi in cui ogni pixel è realizzato tramite una microstrip di silicio (rappresentata in figura 1.2 a destra dal blocco silicon sensor) collegata a un relativo ASIC (*Application-Specific Integrated Circuit*) per consentirne il readout (blocco readout chip nella stessa figura). Quindi un chip ibrido, complessivamente, è costituito da un modulo in cui il chip contenente i pixel è collegato, tramite *bump-bonds connections*, ad un chip con l'elettronica di lettura affacciato ad esso. Il vantaggio principale dei chip ibridi è legato al fatto che la matrice di pixel e l'elettronica di lettura associata a ciascun pixel sono delle componenti indipendenti, e quindi è possibile scegliere il circuito integrato di lettura in modo ottimale in base alle condizioni in cui si richiede al chip di operare. Inoltre, il fatto che ogni pixel viene letto singolarmente e in parallelo rispetto agli altri ha permesso di guadagnare molto in termini di velocità di readout rispetto ai CCD. Gli svantaggi invece comprendono un consumo di potenza elevato, elevati costi di assemblaggio e il grande utilizzo di materiale (da notare, sempre nella figura 1.2, come lo spessore complessivo degli Hybrid Chips sia molto maggiore rispetto ai CCD e ai MAPS), aspetto non ottimale soprattutto se considerato nell'ambito di un tracker.

I chip di tipo **MAPS**, invece, hanno un'architettura simile ai CCD, con la fondamentale differenza che il segnale corrispondente a ciascun pixel viene generato da un circuito integrato all'interno del pixel stesso (*in-pixel electronics*). Questo ha permesso di risolvere per la maggior parte sia i problemi legati ai CCD che agli Hybrid Chips, dal momento che una maggiore tolleranza alla radiazione nei MAPS viene garantita dal fatto che la carica responsabile della generazione del segnale non abbandona mai il pixel di raccolta, mentre grazie all'integrazione dell'elettronica per il readout all'interno del pixel stesso è possibile utilizzare dei layer di silicio molto più sottili, andando quindi a ridurre complessivamente l'utilizzo di materiale per il tracker. Inoltre, sempre grazie alla in-pixel electronics, è possibile ottenere delle alte velocità di lettura, dato che essa viene effettuata in modalità *sparsified*, che consiste nel leggere e trasferire al di fuori del chip solamente l'output di pixel che contengono dei dati, ignorando quindi tutti quelli inattivi.

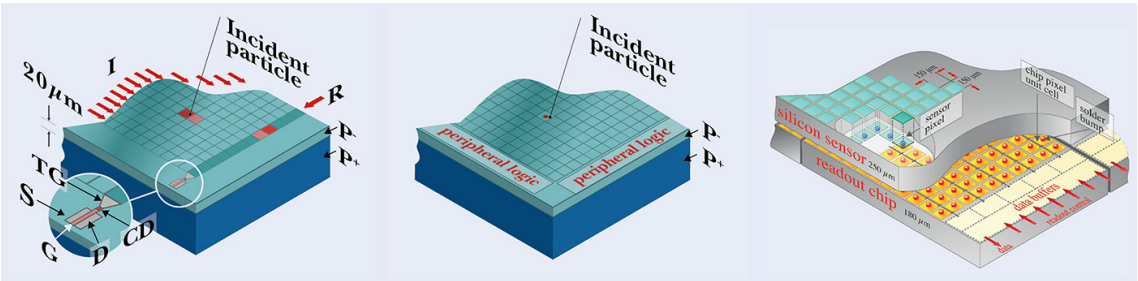

Figura 1.2: Schema di un CCD (a sinistra), di un MAPS (al centro) e di un Hybrid Chip (a destra). In riferimento allo schema del CCD, I indica la zona di imaging, R il readout register, TG il transfer gate, CD il collection diode e infine S, G, D rappresentano i terminali del transistor.

1.2 Il chip ALPIDE

1.2.1 Architettura

Il chip ALPIDE (figura 1.3) dispone di 512×1024 pixel distribuiti su un'area di $15\,\text{mm} \times 30\,\text{mm}$ e comprende anche una regione periferica di area $1.2\,\text{mm} \times 30\,\text{mm}$ che contiene l'elettronica per il readout e il controllo del chip [3].

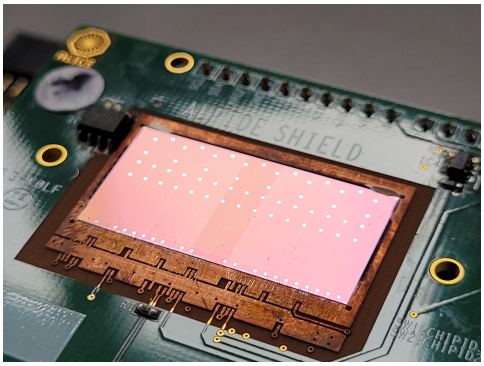

Figura 1.3: Fotografia del chip ALPIDE.

Struttura dei pixel

Il volume attivo è costituito da un layer di silicio p-type (indicato come epitaxial layer P- in figura 1.4) di spessore pari a $25\,\mu\text{m}$ ottenuto tramite crescita epitassiale a partire da un wafer CMOS (substrate P++) [4]. La carica elettrica generata da una particella che attraversa il volume attivo viene raccolta da un array di giunzioni p-n a polarizzazione inversa, con un potenziale positivo di $\sim 1\,\text{V}$ applicato all'n-well diode e un potenziale nullo o negativo, da $0\,\text{V}$ a $-6\,\text{V}$, applicato all'epitaxial layer. La possibilità di modulare tale d.d.p. consente di aumentare la dimensione della zona di svuotamento e, quindi, di diminuire il *charge-collection time*. Il valore standard della d.d.p. è $4\,\text{V}$ e garantisce un charge-collection time inferiore a $15\,\text{ns}$.

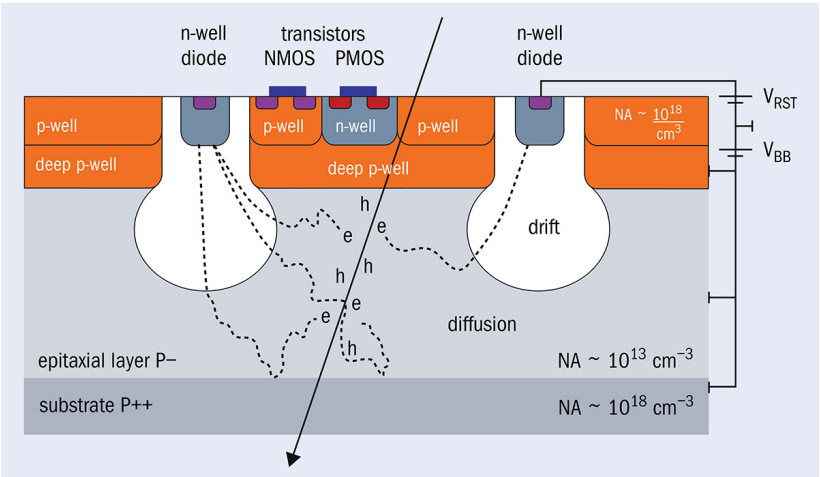

Figura 1.4: Schema della sezione interna di un chip ALPIDE.

Nel momento in cui una particella rilascia una scia di ionizzazione all'interno del layer epitassiale, l'alta concentrazione di elettroni fa sì che essi diffondano verso la zona di svuotamento, ove il campo elettrico ivi presente comporta il loro drift verso l'n-well, per poi essere introdotti nell'in-pixel electronics. Un'ulteriore zona con un drogaggio di tipo p

(deep p-well in figura) ha la funzione di schermare l'n-well contenente il circuito integrato del pixel, in modo da evitare fenomeni di *punch-through*, in cui avviene che della carica spuria interagisce con l'elettronica andando quindi a inficiare l'accuratezza del segnale in uscita dal pixel.

La carica raccolta nell'n-well viene convertita in un segnale digitale da una catena elettronica (figura 1.5) costituita da un preamplificatore, uno shaper, un discriminatore e una *digital section* comprendente, a sua volta, tre *hit storage register* (denominati Multi Event Buffer), un *pixel masking register* e una *pulsing logic*. Il segnale analogico in output dallo shaper raggiunge il massimo in circa 2 μs (*peaking time*) mentre il segnale in uscita dal discriminatore ha una durata media di 10 μs.

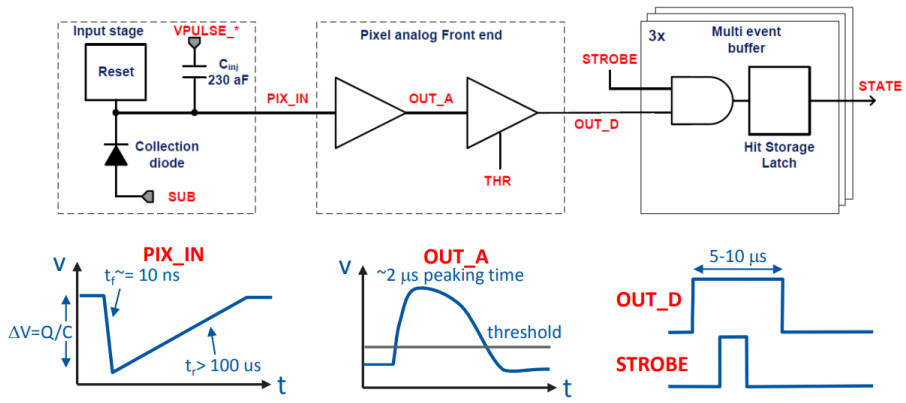

Figura 1.5: Schema a blocchi della catena elettronica all'interno di un pixel.

Struttura della matrice

Al fine di gestire il *readout* della matrice - ovvero la trasmissione dei dati dalla matrice ai circuiti periferici del pixel - e la trasmissione dei dati al di fuori del chip, la matrice di pixel di ALPIDE è suddivisa in **32 regioni** (figura 1.6), numerate da 0 a 31, ciascuna contenente **16 Priority Encoders**, che consistono in circuiti - collegati a 2 colonne consecutive di pixel - che fisicamente trasferiscono i dati ai registri periferici. Specificare l'indirizzo di una regione (REGIONID), il numero di un Priority Encoder (ENCODERID) e l'indirizzo di un pixel (ADDR) rispetto al Priority Encoder considerato equivale a specificare le coordinate X e Y del pixel.

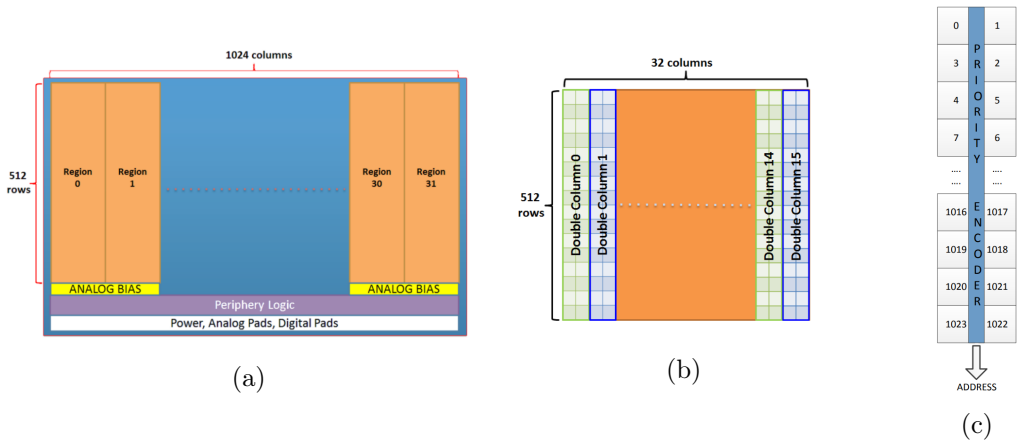

Figura 1.6: (a): suddivisione della matrice in 32 regioni. (b): suddivisione di ciascuna regione in doppie colonne, a cui corrisponde un Priority Encoder. (c): numerazione dei pixel all'interno di ogni priority encoder.

La corrispondenza tra gli indirizzi REGIONID, ENCODERID e ADDR e le coordinate X, Y è espressa dalle seguenti relazioni:

$$X = \begin{cases} 32 \cdot R_{id} + 2 \cdot E_{id} & (I_{px} = 2h \wedge Y = 2k) \vee (I_{px} = 2h+1 \wedge Y = 2k+1) \\ 32 \cdot R_{id} + 2 \cdot E_{id} + 1 & (I_{px} = 2h \wedge Y = 2k+1) \vee (I_{px} = 2h+1 \wedge Y = 2k) \end{cases} \tag{1.1}$$

$$Y = \lfloor I_{px}/2 \rfloor \qquad h,\, k \in \mathbf{N} \tag{1.2}$$

ove con R_{id}, E_{id} e I_{px} si intendono rispettivamente il REGIONID, ENCODERID e ADDR del pixel specificato.

1.2.2 Interfaccia e protocollo di controllo

L'interfaccia e il protocollo di controllo di ALPIDE hanno la funzione di permettere all'utilizzatore di leggere e scrivere sui registri interni del chip, in modo da impostare i parametri di funzionamento, gestire le memorie e distribuire dei segnali che attivano delle funzionalità specifiche. Ogni registro del chip è identificato da un indirizzo (REG ADDR) lungo 16 bit e ha una dimensione di 16 bit.

Protocollo di comunicazione di controllo

Il protocollo di comunicazione di controllo si basa sullo scambio di *characters* lunghi 10 bit aventi la seguente struttura:

- start bit: `0` logico
- messaggio di 8 bit (*payload*)
- end bit: `1` logico

Si definisce *transaction* una sequenza predefinita di characters che permette di compiere un'azione specifica sul chip. Il protocollo di comunicazione prevede 5 tipi di transaction: **BROADCAST COMMAND**, **TRIGGER COMMAND**, **UNICAST WRITE**, **MULTICAST WRITE**, **READ**. Ai fini del lavoro svolto, sono state utilizzate solamente le transaction TRIGGER COMMAND (figura 1.7), UNICAST WRITE (figura 1.8) e READ (figura 1.9), che permettono rispettivamente di inviare un segnale TRIGGER al chip, scrivere un messaggio di 16 bit su un registro e leggere il contenuto di 16 bit di un registro.

Ogni transaction inizia con un **OPCODE character**, in cui gli 8 bit di payload contengono un codice OPCODE che identifica univocamente la transaction. Vi sono inoltre degli OPCODE che non sono associati a una transaction, ma che se inviati al chip all'interno del corrispettivo character permettono di distribuire dei segnali di test o forzare dei reset. Tra gli OPCODE di questa tipologia, è stato utilizzato quello associato al reset globale del chip.

OPCODE	Valore esadecimale	Funzione
TRIGGER	B1	Invia segnale TRIGGER
TRIGGER	55	Invia segnale TRIGGER
TRIGGER	C9	Invia segnale TRIGGER
TRIGGER	2D	Invia segnale TRIGGER
GRST	D2	Reset globale
WROP	9C	Inizia transaction UNICAST WRITE
RDOP	4E	Inizia transaction READ

Tabella 1.1: OPCODE utilizzati e relativa descrizione.

Nel momento in cui non vi è nessuna transaction in corso (*idle state*) tra chip e off-chip hardware, il protocollo prevede che al chip vada inviato il valore 1 logico. Una volta che è iniziata una transaction è comunque possibile far intercorrere degli *idle gap* tra character consecutivi.

Un importante character ricorrente nelle transaction utilizzate è quello contentente il CHIPID, ovvero il codice identificativo di 7 bit del chip all'interno dell'ITS2, che dipende sia dal suo collocamento nella gerarchia tra tutti i chip che dalla sua posizione geografica. Nel nostro caso il CHIPID è `0010000`.

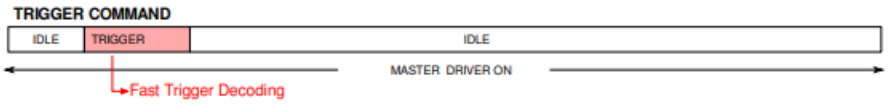

Figura 1.7: Schema di una TRIGGER transaction.

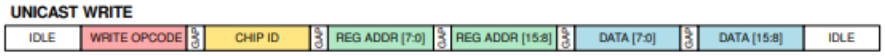

Figura 1.8: Schema di una UNICAST WRITE transaction.

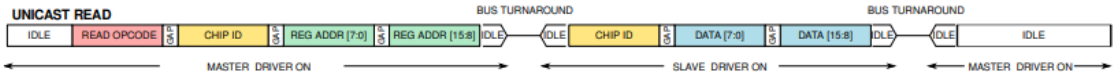

Figura 1.9: Schema di una READ transaction.

Come si può notare nella fig. 1.9, una READ transaction ha una struttura più complessa rispetto alle altre due. Questo perché durante una READ transaction vi è una fase di risposta da parte del chip, che comunica il contenuto (DATA) del registro di cui è si è richiesta la lettura.

Interfaccia di controllo

ALPIDE dispone di due **porte seriali** per implementare le funzioni di controllo: una porta differenziale, denominata DCTRL, e una porta single-ended, denominata CTRL. Quale di queste due porte sia attiva dipende dalla posizione del chip all'interno della gerarchia del tracking system. Per motivi di disponibilità di componenti elettroniche necessarie al funzionamento del chip, si è deciso di operare in modalità **Outer Barrel Master**, che consente di utilizzare la porta DCTRL.

La caratteristica fondamentale delle porte DCTRL e CTRL è che esse supportano una comunicazione **bidirezionale** e ***half-duplex***: ciò significa che i characters scambiati

in direzione hardware-chip viaggiano sulla stessa linea fisica dei characters scambiati in direzione chip-hardware. I dati quindi viaggiano in entrambe le direzioni, ma non contemporaneamente, cosa che se accadesse potrebbe gravemente danneggiare i circuiti interni del chip. Inoltre, la comunicazione tramite la porta DCTRL avviene in modalità sincrona con il **clock** del sistema, che si suppone operi a una frequenza di 40 MHz (frequenza del clock dell'LHC).

Il protocollo di comunicazione prevede che attraverso la porta seriale DCTRL ciascun character venga trasmesso a partire dal bit meno significativo (*LSB first*).

1.2.3 Memorizzazione degli eventi e readout

La funzionalità più importante di ALPIDE è memorizzare e gestire lo stato di ciascun pixel per poi trasmettere l'informazione contenuta in tutta la matrice al di fuori del chip.

L'insieme degli stati di tutti i pixel in un determinato istante si definisce ***frame***, e l'intervallo di tempo in cui il frame viene generato si definisce ***framing window***. In ALPIDE la framing window coincide con l'intervallo di tempo in cui è attivo uno dei tre segnali interni **STROBE**: se tale segnale è attivo e il discriminatore di un qualsiasi pixel si trova al di sopra del livello di threshold impostato, allora un evento (*hit*) verrà memorizzato in uno dei tre registri interni di ciascun pixel. L'insieme dei tre registri interni di cui dispone ogni pixel viene denominato Multi Event Buffer (MEB). ALPIDE, quindi, è in grado di tenere in memoria tre frame completi senza perdita di informazione. Ciascuno dei tre registri del MEB, indicizzati con le lettere A, B e C, può essere scritto, o sovrascritto, solamente in seguito al completamento della framing window del corrispettivo segnale STROBE_A, STROBE_B, STROBE_C. I segnali STROBE non possono essere attivati direttamente dall'off-chip hardware, bensì vengono attivati automaticamente dal chip in seguito alla ricezione, tramite l'interfaccia di controllo, di un character TRIGGER.

Il ***readout*** di ciascun frame, ovvero la trasmissione dell'informazione contenuta nei MEB al di fuori della matrice, è governato da altri tre segnali interni denominati MEMSEL. Tali segnali vengono attivati automaticamente in seguito alla disattivazione del corrispettivo segnale STROBE: è quindi possibile il readout di un solo frame alla volta. La sequenza di attivazione di tutti i segnali relativi al framing e al readout è riportato nella figura 1.10.

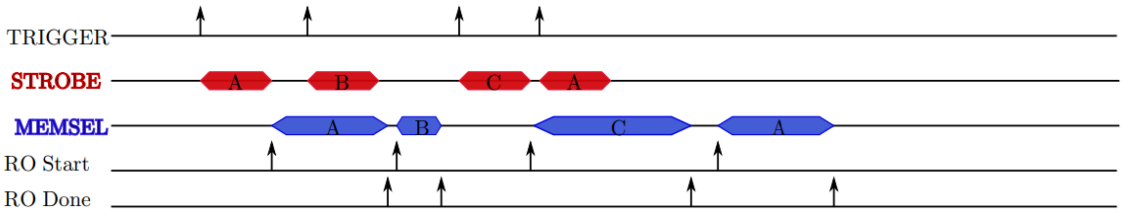

Figura 1.10: Schema del timing del readout.

ALPIDE può operare in due modalità che prevedono una gestione diversa del readout:

- **Trigger Mode**. Tale modalità è pensata per i contesti in cui è necessario triggerare il chip in istanti ben precisi e attivare delle framing window di durata ridotta (i segnali STROBE rimangono attivi per qualche centinaio di nanosecondi). Il readout avviene in modo da dare la priorità ai frame già memorizzati nei MEB: nel caso in cui i tre registri dei MEB siano già pieni, la ricezione di un nuovo TRIGGER darà come risultato un *empty chip data packet*, in modo da terminare il readout in corso senza perdita di infomazione.

- **Continuous Mode**. Tale modalità è pensata per i contesti in cui è necessario triggerare periodicamente il chip, attivando delle framing window di durata maggiore (i

segnali STROBE rimangono attivi per pochi microsecondi). In questo caso l'intervallo di tempo tra la fine di uno STROBE e l'attivazione del successivo è di fatto un **dead time**, e quindi deve essere mantenuto il più piccolo possibile (il tempo raccomandato è 100 ns). Il readout avviene in modo da dare la priorità alle nuove richieste di framing: nel caso in cui due dei tre registri dei MEB siano già pieni, la ricezione di un nuova richiesta di framing andrà a sopprimere il readout in corso, comportando quindi una perdita di informazione, in modo da garantire che vi sia sempre un registro del MEB libero per la richiesta di framing successiva.

Nel nostro caso, ALPIDE è stato impostato in Continuous Mode. Inoltre, ci si è serviti dell'**Internal Sequencer** di ALPIDE, programmabile tramite l'interfaccia di controllo, che consente di generare degli STROBE periodici, e il relativo gap temporale tra di essi, in modo automatico in seguito alla ricezione di un singolo TRIGGER iniziale.

1.2.4 Interfaccia e protocollo di trasmissione dei dati

L'interfaccia e il protocollo di trasmissione hanno la funzione di trasmettere i dati raccolti in una framing window all'off-chip hardware.

Protocollo di trasmissione

Il protocollo di trasmissione si basa sull'invio da parte del chip di una serie di **data words**.

Data Word	No. bit	Valore binario
IDLE	8	1111_1111
CHIP HEADER	16	1010<chip id[3:0]><BUNCH COUNTER FOR FRAME[10:3]>
CHIP TRAILER	8	1011<readout flags[3:0]>
CHIP EMPTY FRAME	16	1110<chip id[3:0]><BUNCH COUNTER FOR FRAME[10:3]>
REGION HEADER	8	110<region id[4:0]>
DATA SHORT	16	01<encoder id[3:0]><addr[9:0]>
DATA LONG	24	00<encoder id[3:0]><addr[9:0]> 0 <hit map[6:0]>
BUSY ON	8	1111_0001
BUSY OFF	8	1111_0000

Tabella 1.2: Data Word previste da ALPIDE.

La struttura di un messaggio del chip per la trasmissione i dati raccolti durante una framing window ha la seguente struttura:

1. **CHIP HEADER** word: segna l'inizio di una trasmissione dati. Contiene gli ultimi 4 bit meno significativi del CHIPID che identifica il chip e gli 8 bit più significativi del contatore interno BUNCH COUNTER FOR FRAME che fornisce un timestamp per la ricostruzione temporale della sequenza di ricezione dei frame. Ai fini del lavoro svolto, non è stato necessario tenere conto del valore assunto da tale contatore.

2. **REGION HEADER** word: contiene i 5 bit che codificano il REGIONID, ovvero l'indirizzo che identifica ciascuna delle 32 regioni in cui è divisa la matrice con un numero da 0 a 31.

3. **DATA SHORT** o **DATA LONG** word. La word DATA SHORT comunica che un determinato pixel ha registrato una hit. Essa contiene gli indirizzi ENCODERID e ADDR che identificano rispettivamente il numero (da 0 a 15) del priority encoder all'interno della regione precedentemente specificata a cui è collegato il pixel e l'indirizzo del pixel all'interno della doppia colonna associata al priority encoder. La word DATA LONG comunica che un determinato cluster di 8 pixel contiene delle hit. Essa

viene trasmessa solamente se, tramite l'interfaccia di controllo, è stata abilitata la funzionalità di clustering. Viene specificata la locazione del primo pixel del cluster tramite gli stessi indirizzi presenti nella word DATA SHORT, mentre lo stato dei restanti 7 pixel viene trasmesso in una HITMAP.

4. **CHIP TRAILER** word: segna la fine di una trasmissione dati. Essa contiene 4 bit di READOUT FLAGS che rappresentano degli avvisi riguardanti lo stato interno del chip.

Nel momento in cui non vi è una trasmissione di dati in corso, il chip restituisce la data word IDLE. Tale data word può presentarsi anche una volta iniziata la trasmissione dei dati tra una data word e la successiva, a meno che esse non abbiano una lunghezza di 16 o 24 bit. Le data word BUSY ON e BUSY OFF vengono trasmesse dal chip, tra due word qualsiasi, nel momento in cui il chip è vicino alla saturazione della sua capacità di processare i dati raccolti. Nel caso in cui si operi in Continuous Mode, BUSY ON viene trasmesso nel momento in cui il chip riceve una richiesta di framing e uno dei tre registri dei MEB è ancora occupato. BUSY OFF viene trasmesso quando la condizione di attivazione di BUSY ON viene meno. Infine la data word CHIP EMPTY FRAME viene trasmessa quando all'interno di una framing window non sono state registrate delle hit in alcun pixel. Essa contiene gli stessi campi presenti nella word CHIP HEADER.

Interfaccia di trasmissione

ALPIDE dispone di due porte fisiche per la trasmissione dei dati all'off-chip hardware: una porta seriale differenziale HSDATA e una porta parallela a 4 bit DATA[3:0]. Avendo impostato il chip in modalità Outer Barrel Master, è possibile utilizzare entrambe le porte, ma per ragioni di disponibilità di componenti elettroniche si è deciso di adoperare la porta parallela.

La porta parallela opera in modo sincrono con il clock del sistema (40 MHz), e di default trasmette in **Double Data Rate** (DDR). Ciò significa che vengono trasmessi 4 bit sul *positive edge (posedge)* e altrettanti sul *negative edge (negedge)* del clock. In totale si ottiene una velocità di trasmissione pari a $40 \cdot 2 \cdot 4 = 320\,\mathrm{Mb/s}$.

Il protocollo di comunicazione prevede che ALPIDE trasmetta i 4 bit più significativi di ciascun byte sempre sul posedge del clock.

Capitolo 2

Sviluppo del firmware

Al fine di mettere in funzione ALPIDE e analizzare i dati raccolti dai pixel è necessario realizzare un circuito esterno ad esso (*off-chip hardware*) che lavori in simbiosi con il chip stesso. Tale circuito, quindi, dovrà essere in grado sia di comunicare con il chip attraverso l'interfaccia di controllo sia di raccogliere i dati in uscita dall'interfaccia di trasmissione per inviarli al PC, ove essi verranno elaborati per essere resi maggiormente fruibili da parte dell'utilizzatore.

L'insieme di istruzioni necessarie all'off-chip hardware per operare correttamente prendono il nome di **firmware**. L'off-chip hardware e il firmware vengono realizzati grazie a un ***Hardware Description Language* (HDL)**, ovvero un linguaggio che permette, tramite una sintassi simile ai comuni linguaggi di programmazione, di descrivere le funzionalità dell'hardware che si vuole implementare lasciando però che l'effettiva sintesi, in termini di componenti fisici, avvenga tramite delle funzionalità automatizzate. Nel nostro lavoro, è stato utilizzato il linguaggio Verilog [5].

Il supporto fisico su cui viene realizzato l'off-chip hardware è una ***Field Programmable Gate Array* (FPGA)**, ovvero un dispositivo che contiene al suo interno un gran numero di componenti digitali. Essendo tale dispositivo programmabile, queste componenti possono essere collegate tra di loro in base alle delle istruzioni fornite da un codice HDL, in modo da realizzare un circuito modificabile che implementa una o più funzioni specifiche. L'FPGA utilizzata nel nostro lavoro è una Artix-7 35T Arty [6], scelta perché permette un collegamento Ethernet con il PC dal momento che è provvista della corrispettiva porta.

2.1 Overview del firmware

Il firmware necessario per il funzionamento dell'FPGA è stato realizzato all'interno dell'IDE *Xilinx Vivado 2021.2* [7], sviluppato da AMD. Esso consente di gestire il workflow da inizio a fine: a partire dai blocchi di codice scritti in HDL, Vivado è in grado di generare una serie di istruzioni per l'FPGA affinché implementi al suo interno il circuito desiderato.

Come mostrato nella figura 2.1, lo sviluppo del firmware avviene tramite la scrittura di *moduli*, ovvero blocchi di codice HDL che realizzano una o più operazioni. Nella definizione di un blocco, è necessario specificare gli *input* e gli *output*, che non corrispondono propriamente a delle variabili, bensì a dei collegamenti fisici (*wire*) o a delle posizioni di memoria (*registers*) i cui valori vengono modificati dal codice costituente il corpo del modulo. Gli input/output dei vari moduli possono essere collegati tra loro per creare dei moduli più grandi. Infatti, sempre in riferimento alla figura 2.1, il firmware sviluppato si articola in 2 macro-moduli (`dctrl manager` e `readout deserializer`) che gestiscono separatamente l'interfaccia di controllo e l'interfaccia di trasmissione dei dati di ALPIDE. Il modulo `dctrl manager` è composto da 4 moduli (`read`, `initialize`, `write` e `trigger`) ognuno dei quali implementa una specifica funzione che verrà illustrata nella prossima sezione. Infine `dctrl`

`manager`, assieme al modulo `readout deserializer`, compone il modulo `top`, comunemente chiamato così in questi casi perché è il modulo più esterno che complessivamente racchiude tutti i sottoblocchi del circuito implementato sull'FPGA.

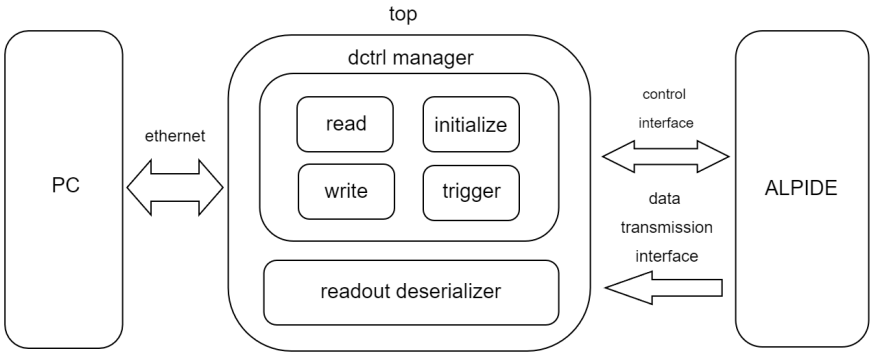

Figura 2.1: Rappresentazione schematica del firmware implementato nell'FPGA (blocco centrale) con i relativi moduli.

Una caratteristica comune a tutti i moduli è che essi sono stati pensati e scritti come **macchine a stati**: ciò significa che le operazioni eseguite da ciascun modulo dipendono dallo stato interno del modulo stesso. Lo stato del modulo cambia in base al valore assunto da alcuni registri o wire che vengono stabiliti dall'utente. Nel corso dei paragrafi successivi verranno richiamati gli stati relativi al modulo discusso qualora presentassero degli aspetti particolarmente rilevanti, dal momento che riportare nel dettaglio il funzionamento di ogni singolo modulo appesantirebbe la trattazione e non costituirebbe del materiale significativo da includere in questa tesi.

2.2 Firmware per l'interfaccia di controllo

Il firmware realizzato per il funzionamento dell'interfaccia di controllo si articola in 4 moduli, `initialize`, `read`, `write` e `trigger`, che assieme costituiscono il modulo `dctrl manager`. L'insieme di questi moduli ha il compito di gestire il dialogo tra FPGA e ALPIDE attraverso l'interfaccia di controllo, seguendo il protocollo esposto nel capitolo precedente.

2.2.1 Modulo initialize

Il modulo `initialize` è deputato all'impostare i principali registri interni del chip necessari al suo funzionamento e, pertanto, è pensato per essere attivato nel momento dell'avvio del chip. Lo stato `IDLE` del modulo è quello in cui non vi sono scambi di transactions tra FPGA e chip. L'FPGA invia al chip una sequenza di 1 logici come previsto dal protocollo di comunicazione. Nel momento in cui il segnale `INIT_ENABLE` (figura 2.2) viene inviato dal PC, il modulo entra nello stato `SETTING_STAGES`: in questo stato il modulo prima effettua un reset globale del chip, inviando un GRST character, e successivamente invia al chip una sequenza di WRITE transactions attraverso l'output `init_dctrl` per impostare tutti i registri necessari. In particolare, viene impostata in Continuous Mode la modalità di readout, viene fissata a 5 μs la durata dei segnali STROBE e a 100 ns la durata del gap tra un segnale STROBE e il successivo. Al termine di tutte le transactions previste, viene attivato il segnale `INIT_COMPLETED` in modo da comunicare al PC che l'inizializzazione è completata. Il modulo quindi passa allo stato `WAIT_INIT_ENABLE_OFF`, in cui si attende che il PC disattivi il segnale `INIT_ENABLE`, in modo da far tornare il modulo nello stato

`IDLE`.

Nella dichiarazione del modulo sono presenti altri segnali: l'input `clock` rappresenta il clock del sistema ed è necessario includerlo dal momento che, come riportato in precedenza, le control transactions avvengono in modalità sincrona. L'output `init_dctrl` rappresenta il canale fisico attraverso il quale le control transactions inviate durante lo stato `SETTING_STAGES` vengono inoltrate al chip. Infine gli output `isINIT_ENABLE`, `isWAIT_INIT_ENABLE_OFF` e `isINIT_ENABLE` servono solo come segnali di avviso e/o controllo per gli altri moduli contenuti nel `dctrl manager`.

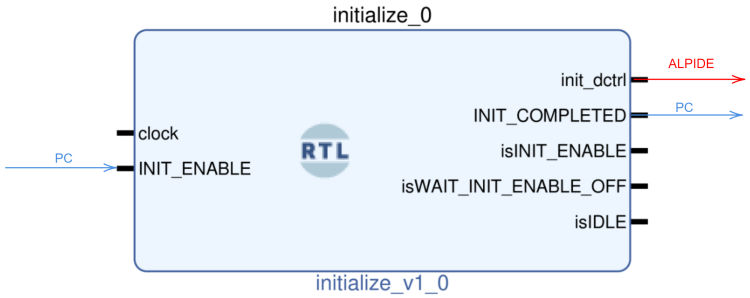

Figura 2.2: Block design del modulo `initialize`.

2.2.2 Modulo read

Il modulo `read` è deputato alla lettura dei registri interni del chip. Esso è un modulo particolarmente complesso dal momento che durante la lettura di un registro è prevista una fase di risposta da parte del chip - in cui viene trasmesso il contenuto del registro richiesto - che avviene sulla stessa linea fisica in cui viene inoltrata la richiesta di lettura da parte dell'FPGA. Il firmware, quindi, dovrà garantire che il segnale in direzione hardware-chip non si sovrapponga con il segnale in direzione chip-hardware. A tal proposito, viene specificato nella documentazione di ALPIDE il timing che deve sussistere tra i due segnali (figura 2.3).

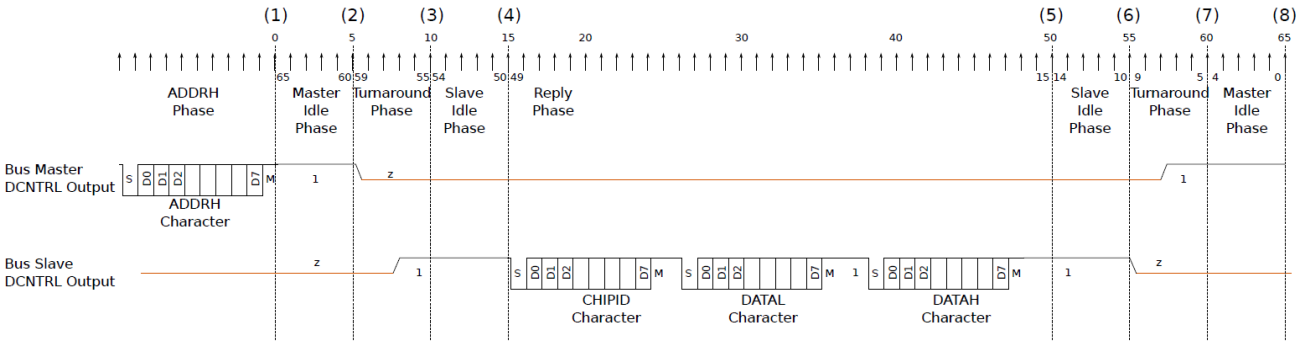

Figura 2.3: Schema della fase di replica da parte del chip in seguito a una richiesta di lettura. In alto, le freccette rappresentano i posedge del clock, mentre al di sopra di esse è riportato il conteggio dei cicli di clock che scandiscono le varie parti della comunicazione.

Sempre in riferimento alla figura, per *bus master* si intende la linea fisica utilizzata dall'FPGA per inoltrare i characters in direzione chip, mentre per *bus slave* si intende la linea fisica utilizzata da ALPIDE per inoltrare le transaction in direzione FPGA. Dal punto di

vista fisico le due linee coincidono, ma vengono raffigurate separatamente per chiarezza. Il funzionamento del modulo è il seguente: dopo la ricezione di una richiesta di lettura da parte del PC (segnale `ADDR_RECEIVE` in figura 2.4), l'FPGA compone una READ transaction - in accordo con il protocollo di comunicazione - che viene successivamente inoltrata al chip. Infatti, nell'immagine sopra, si nota come i primi 10 cicli di clock corrispondano al character in cui vengono comunicati al chip gli 8 bit più significativi del registro di cui si vuole leggere il contenuto (confronta con la figura 1.9). Successivamente, il protocollo prevede che vi siano 5 cicli di clock in cui viene comunicato al chip il valore 1 logico (Master Idle Phase), a cui segue la prima **Turnaround Phase**: nei 5 cicli di clock ad essa dedicati la linea fisica dell'FPGA assume uno stato logico chiamato `Z` (**alta impedenza**), grazie al quale l'FPGA di fatto cede il controllo della linea stessa ad ALPIDE. Una volta avvenuto ciò, il chip replica con una Slave Idle Phase - del tutto analoga a quella menzionata poco fa - per poi comunicare all'FPGA il contenuto del registro indicato tramite tre character. Infine, successivamente a un'altra Slave Idle Phase, vi è la seconda Turnaround Phase, in cui l'FPGA riprende il controllo della linea fisica.

Dal punto di vista del firmware, ciò che avviene durante una Turnaround Phase viene realizzato tramite una porta logica **tri-state**, ovvero una porta logica che soddisfa la seguente tabella di verità:

in	ts	out
0	0	0
1	0	1
0	1	Z
1	1	Z

Tabella 2.1: Tabella di verità di una porta tri-state.

Si noti come il bit `ts` funge da *enable*: se esso è uguale a 0 allora la porta logica corrisponde all'identità, mentre se è uguale a 1 allora il valore di `in` viene ignorato, dal momento che la porta si aspetta, al contrario, di ricevere "in input" un bit dal suo bit di output. Tuttavia, nella scrittura del modulo `read` (figura 2.4), sono stati dichiarati due wire distinti (`read_dctrl_in` e `read_dctrl_out`) per le due direzioni di comunicazione - chip-FPGA e FPGA-chip - uno come input e l'altro come output: questo perché è buona prassi implementare i vari moduli in modo che essi realizzino solamente la funzionalità specifica per cui vengono pensati, senza tenere conto di complicazioni contingenti come quella, nel nostro caso, di avere a che fare con una linea di comunicazione half-duplex. La porta logica tri-state, quindi, viene realizzata solamente in ultima istanza all'interno del modulo `top` e il ruolo del bit `ts` riportato nella tabella 2.1 viene svolto da un segnale in output dal modulo `read` (`isDATA_RECEIVE`). Ciò che si ha nel modulo `top` è riportato nel codice 2.1: il wire `t_dctrl`, che corrisponde all'unica linea fisica realmente presente tra l'FPGA e ALPIDE, è dichiarato come un *inout*, ovvero un wire che si può comportare sia da input che da output in base al valore del segnale `handle_dctrl_ts` [1] e viene assegnato, tramite un *ternary operator*, al valore `1'bZ` nel caso in cui `handle_dctrl_ts` sia uguale al valore 1 logico o al valore assunto da `handle_dctrl_out` nel caso in cui `handle_dctrl_ts` sia uguale al valore 0 logico. Contestualmente, si nota che il segnale `handle_dctrl_in` viene assegnato al valore di `t_dctrl`: in questo caso, trattandosi di un'assegnazione semplice e siccome `t_dctrl` si trova a destra dell'operatore di assegnazione, il valore di `handle_dctrl_in` verrà modi-

[1] corrispondente al segnale `isDATA_RECEIVE` menzionato poco prima, solamente che nel modulo `top` è stato rinominato per chiarezza. Anche i segnali `read_dctrl_out` e `read_dctrl_in`, richiamati poco dopo, sono stati rinominati nel modulo `top`.

ficato solamente quando `t_dctrl` opera come input, ovvero quando sta trasmettendo in direzione chip-FPGA.

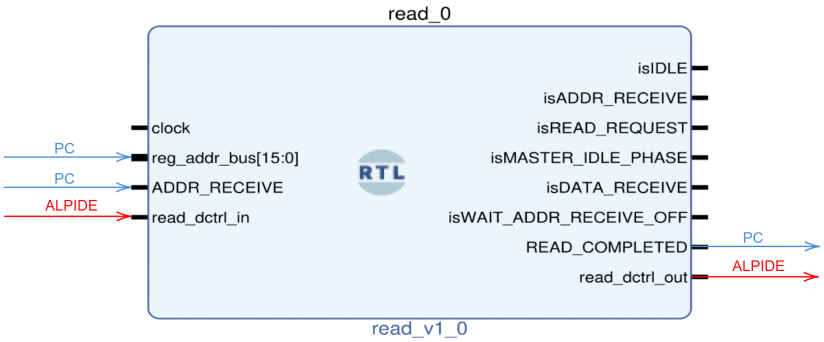

Figura 2.4: Block design del modulo `read`.

```
1    assign t_dctrl = handle_dctrl_ts ? 1'bZ : handle_dctrl_out;
2    assign handle_dctrl_in = t_dctrl;
```

Codice 2.1: Implementazione della porta tri-state tramite un ternary operator (prima riga) nel modulo `top`.

2.2.3 Modulo write

Il modulo `write` ha la funzione di sovrascrivere l'informazione contenuta all'interno di un particolare registro del chip. La struttura del modulo è rappresentata in figura 2.5; esso riceve in input dal PC il segnale di richiesta di scrittura (`WRITE`), l'indirizzo del registro su cui scrivere (`REG_ADDR[15:0]`) e il contenuto da scrivere sul registro indicato (`WRITE_DATA[15:0]`). La write transaction viene inviata ad ALPIDE tramite il wire `write_dctrl` e il completamento della transaction viene segnalato al PC tramite il wire `WRITE_COMPLETED`.

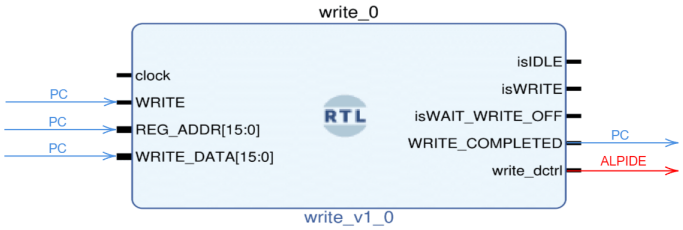

Figura 2.5: Block design del modulo `write`.

2.2.4 Modulo trigger

Il modulo `trigger` ha la funzione di inviare al chip un TRIGGER character in modo da dare inizio all'attivazione ciclica dei segnali STROBE. Come si può vedere in figura 2.6, il modulo riceve in input un segnale di richiesta di trigger (`TRIGGER`) da parte del PC,

compone il TRIGGER character e lo invia al chip tramite il wire `trig_dctrl`. Infine, notifica al PC l'avvenuto invio del character tramite il wire `TRIGGER_COMPLETED`.

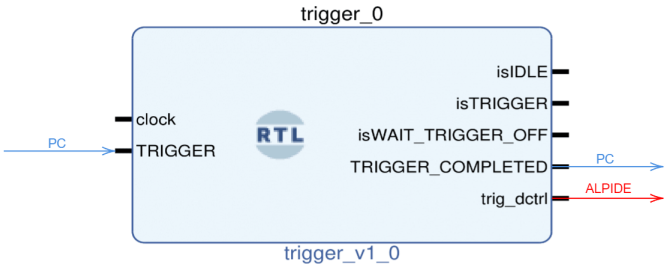

Figura 2.6: Block design del modulo `trigger`.

2.3 Firmware per l'interfaccia di trasmissione dei dati

Il firmware per l'interfaccia di trasmissione dei dati ha il compito di prendere in input i dati relativi alle hit registrate da ALPIDE, in uscita dalla porta parallela a 4 bit DATA[3:0], e inoltrarli al PC, dove verranno successivamente elaborati. La struttura di questa parte del firmware è più semplice, dal momento che consiste nel solo modulo `readout deserializer`, chiamato così perché deserializza l'output del chip, trasmettendo al PC 8 bit in parallelo in Single Data Rate (SDR) invece che 4 bit in parallelo in Double Data Rate (DDR). La necessità di implementare questo modulo deriva dal fatto che è più conveniente per il PC operare con 8 bit in SDR, dal momento che questa configurazione si presta meglio per trasferire i dati tramite una porta Ethernet. Inoltre, come mostrato nella tabella 1.2, le data word in output da ALPIDE previste dal protocollo di trasmissione dei dati hanno tutte la lunghezza di un byte o di suoi multipli interi, e quindi maneggiare byte interi invece di metà byte rende più agevole l'elaborazione dell'output di ALPIDE.

Per realizzare il `readout deserializer` ci si è serviti di un modulo contenuto all'interno delle *7 Series FPGAs SelectIO Resources* [8], che è una raccolta di template di codice, realizzato da Xilinx, pensati per implementare delle funzionalità di input/output specifiche per FPGA della serie 7, come quella utilizzata in questo lavoro. In particolare, è stato utilizzato il modulo IDDR (Input DDR), che ha la seguente struttura:

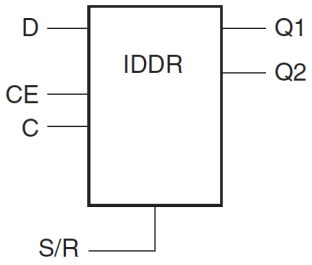

Figura 2.7: Schema del modulo IDDR con i relativi input e output.

Le porte di input sono C, ove va collegato il clock del sistema, il relativo CE (Clock Enable), da tenere sempre al valore 1 logico, e D, che va collegato a una delle linee a singolo bit in DDR che si intende deserializzare: dovendo noi passare da 4 bit a 8 bit saranno necessarie, pertanto, 4 istanze del modulo IDDR. La porta S/R funge da set/reset e per un funzionamento normale del modulo è necessario impostarla al valore 0 logico. Gli output,

invece, sono due linee a singolo bit Q1 e Q2 in cui vengono mappati, rispettivamente, i bit trasmessi sul posedge e sul negedge della linea in input D. I bit Q1 e Q2 vengono trasmessi sempre in modalità sincrona con il clock del sistema, anche se il modulo IDDR consente diverse configurazioni di sincronizzazione: quella scelta da noi prende il nome di *Same Edge Pipelined* e prevede che i due bit trasmessi da D in un ciclo di clock vengano trasmessi simultaneamente da Q1 e Q2 con due cicli di clock di ritardo, come mostrato in figura 2.8. La scelta di utilizzare 4 moduli IDDR non influisce sulla velocità di trasmissione dei dati, dato che anche se si passa da una trasmissione DDR a una SDR la quantità di bit trasmessi per ciclo di clock rimane costante, e quindi la velocità risulta sempre 320 Mb/s.

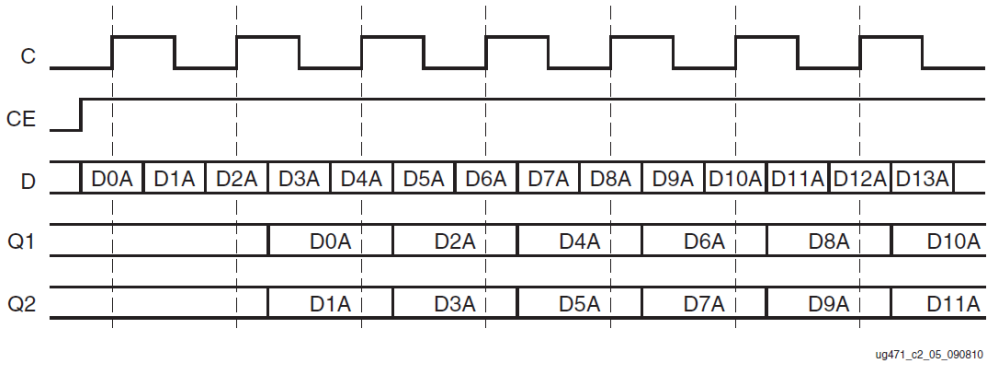

Figura 2.8: Schema del timing del modulo IDDR in modalità Same Edge Pipelined.

Il modulo `readout deserializer`, quindi, riceve in input il segnale DDR a 4 bit paralleli in uscita da ALPIDE (`data_in[3:0]` nella figura 2.9) e, tramite i 4 moduli IDDR istanziati al suo interno, genera il segnale SDR in output `data_out[7:0]`. `readout deserializer` inoltre, è in grado di generare un segnale di output chiamato `valid_data` che ha la funzione di comunicare al PC gli intervalli di tempo esatti in cui deve memorizzare l'output di ALPIDE, in modo da evitare di elaborare grandi quantità di bit privi di informazione contenuti nelle IDLE e CHIP EMPTY FRAME word. Infine, il segnale `busy` viene attivato nei casi specificati nel capitolo precedente.

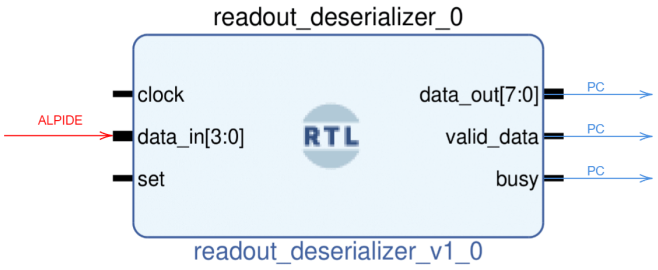

Figura 2.9: Block design del modulo `readout deserializer`.

Capitolo 3

Simulazioni

In questo capitolo si vogliono riportare i risultati più significativi ottenuti da alcune simulazioni che sono state effettuate per verificare la funzionalità del firmware presentato nel capitolo precedente. Relativamente al firmware per l'interfaccia di controllo verrà riportata una simulazione di una richiesta di lettura, mentre per quanto riguarda il firmware per l'interfaccia di trasmissione dei dati verranno proposte due simulazioni di presa dati in condizioni diverse.

La possibilità di includere in questa tesi solamente i risultati di alcune simulazioni piuttosto di dati veri e propri deriva dal fatto che per motivi logistici non è stato possibile avere a disposizione alcune componenti elettroniche necessarie al collegamento di ALPIDE all'FPGA utilizzata. Questa circostanza, pertanto, ha reso necessaria la scrittura di un'ulteriore parte di firmware che simulasse le funzionalità principali di ALPIDE.

3.1 Firmware per la simulazione di ALPIDE

Il firmware necessario per simulare le funzionalità principali di ALPIDE è stato organizzato nei due moduli `ALPIDE dctrl` e `ALPIDE readout`: il primo è responsabile della simulazione di una risposta conseguente a una richiesta di lettura, mentre il secondo ha il compito di simulare l'output del chip in seguito alla rivelazione di radiazione da parte della matrice di pixel. Allo stesso modo di quanto fatto con il firmware per l'FPGA, i due moduli suddetti sono stati inglobati all'interno di un modulo `ALPIDE top`, in modo da raggruppare tutte le porte di input/output in un unico modulo, come mostrato nella figura 3.1.

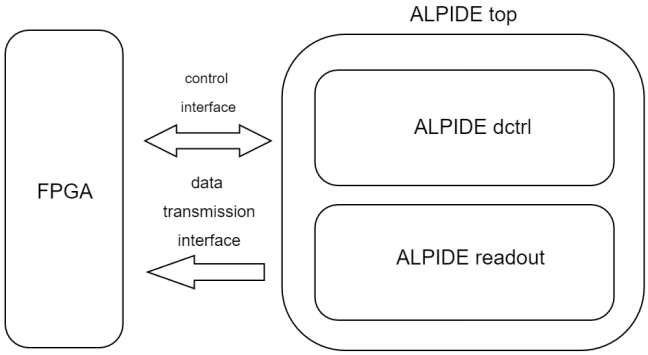

Figura 3.1: Rappresentazione schematica del firmware realizzato per la simulazione di ALPIDE.

Il modulo `ALPIDE dctrl` è anch'esso realizzato come macchina a stati ed è in grado di elaborare una richiesta di lettura in arrivo dall'FPGA tramite il wire `adctrl_in` (figura 3.2), seguendo il timing esposto nella figura 2.3. Questo avviene grazie al segnale `isMASTER_IDLE_PHASE`, sempre proveniente dall'FPGA, che permette di coordinare varie fasi della comunicazione previste. La risposta alla richiesta di lettura viene trasmessa all'FPGA tramite il wire di output `adctrl_out`. Infine, il wire di output `adctrl_ts` serve a implementare, nel modulo `ALPIDE top`, una porta tri-state del tutto analoga a quella esposta per il modulo `read`. L'input `REG_DATA[15:0]`, invece, contiene i due byte di informazione contenuti nel registro (fittizio) che si vuole leggere, e quindi rappresenta un parametro della simulazione.

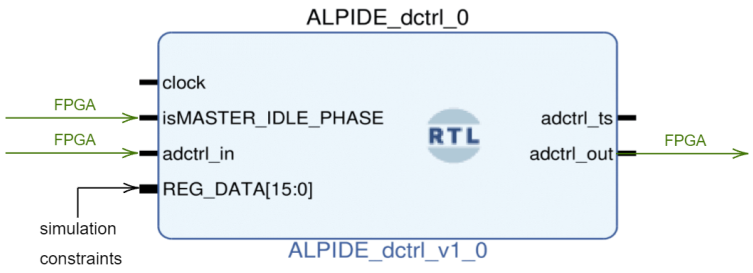

Figura 3.2: Block design del modulo `ALPIDE dctrl`.

Il modulo `ALPIDE readout` è pensato per ricevere in input le coordinate X e Y dei pixel che hanno rivelato una hit, generate casualmente in base a una distribuzione predefinita, per poi trasmetterle all'FPGA tramite la porta parallela DDR a 4 bit `DATA[3:0]` (figura 3.3), seguendo le regole del protocollo di trasmissione dei dati. Le coordinate X e Y vengono fornite in input dal bus di 19 wire `pxl_hit[18:0]`: i primi 10 bit sono riservati alla coordinata X, che prende valori da 0 a 1023, mentre gli ultimi 9 bit sono riservati alla coordinata Y, che prende valori tra 0 e 511. Tuttavia, le coordinate vengono memorizzate solamente quando il segnale in input `strobe` è attivo, mentre la trasmissione inizia nel momento in cui tale segnale viene disattivato (confronta la figura 1.10). La scelta di utilizzare solamente un segnale `strobe` invece dei tre previsti dall'architettura interna di ALPIDE deriva solamente da motivi di semplicità, ma ciò non inficia in alcun modo la validità generale delle simulazioni effettuate. In questo modulo, pertanto, i segnali `strobe` e `pxl_hit[18:0]` costituiscono dei parametri della simulazione.

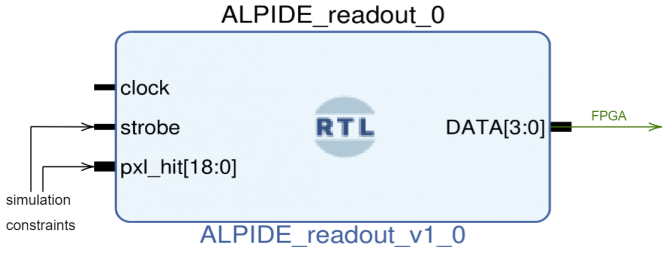

Figura 3.3: Block design del modulo `ALPIDE readout`.

Il block design complessivo del setup per le simulazioni è riportato in figura 3.4. In essa è anche incluso il blocco Clocking Wizard, che consiste in un modulo template che distribuisce il clock del sistema agli altri moduli (il segnale del clock è raffigurato in verde, così come gli altri segnali interni all'FPGA). In blu sono raffigurati i segnali da e per il PC, mentre in rosso sono marcati i segnali scambiati tra FPGA e ALPIDE. Infine in nero sono riportati i segnali corrispondenti ai parametri della simulazione.

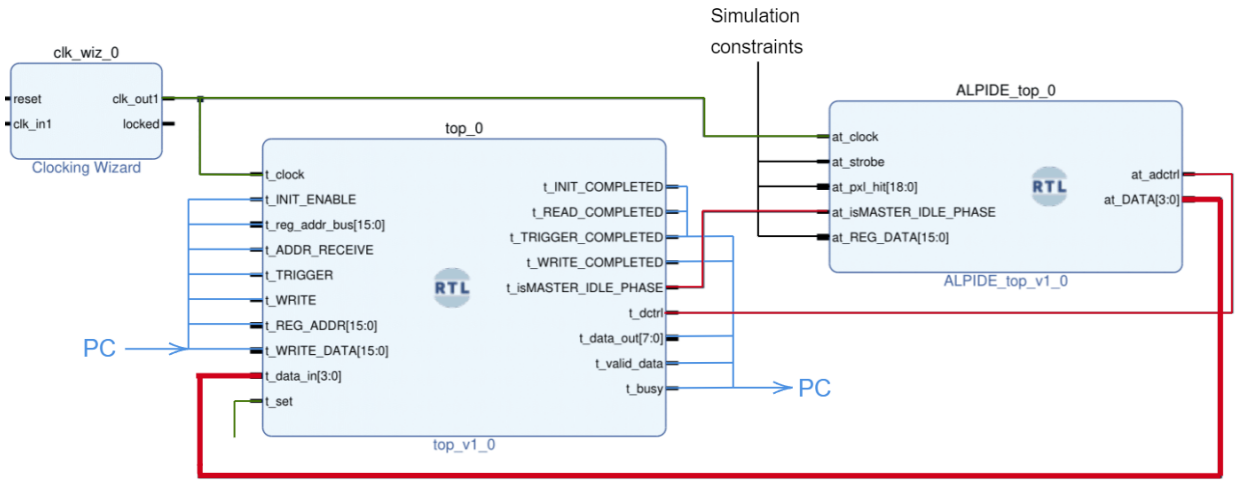

Figura 3.4: Block design complessivo del setup per la simulazione.

3.2 Simulazione richiesta di lettura

La prima simulazione effettuata ha consistito nell'invio da parte dell'FPGA di una richiesta di lettura del registro (fittizio) di indirizzo `16'b1001010011101011`. Il contenuto del registro indicato (`at_REG_DATA[15:0]`) è stato posto uguale a 16 bit nulli, in modo da poter verificare più rapidamente se vi fosse corrispondenza tra il segnale atteso e quello osservato nella simulazione. Una volta avviata la simulazione, dopo 200 ns viene attivato il segnale `g_ADDR_RECEIVE` (visibile nella figura 3.4) che dà inizio alla sequenza di istruzioni contenute nel modulo `read`. In seguito alla risposta da parte di ALPIDE, il segnale `g_ADDR_RECEIVE` viene azzerato dopo poco più di 3 μs dalla sua attivazione. La forma dei segnali al variare del tempo durante la simulazione è riportato nella figura 3.5: in azzurro sono evidenziati i segnali che, in un caso reale, provengono dal PC, in giallo i segnali relativi all'FPGA, in fucsia i segnali relativi al chip e in rosso i segnali scambiati tra FPGA e ALPIDE.

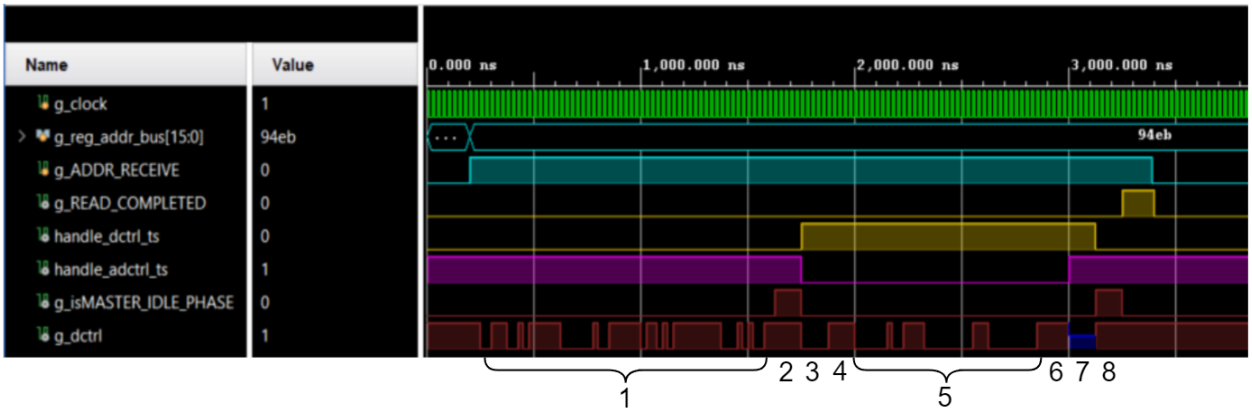

Figura 3.5: Forme dei segnali durante la simulazione di lettura.

Inoltre, in figura sono numerate le varie fasi della comunicazione tra FPGA e chip: (1) corrisponde alla richiesta di lettura trasmessa dall'FPGA ad ALPIDE, a cui segue (2) la prima Master Idle Phase. (3) corrisponde alla prima Bus Turnaround Phase, in cui ALPIDE prende controllo della linea. Successivamente il chip continua con la prima Slave Idle Phase (4) a cui segue la risposta alla richiesta di lettura (5). Infine la comunicazione termina con il susseguirsi della seconda Slave Idle Phase (6), la seconda Bus Turnaround Phase (7) e la seconda Master Idle Phase (8). Notare come durante la seconda Bus Turnaround Phase la linea `g_dctrl` assuma il colore blu, ad indicare che si trova nello stato `Z`, a causa del fatto che i due segnali `handle_dctrl_ts` e `handle_adctrl_ts` sono entrambi uguali a 1 durante quei 5 cicli di clock, a differenza di quanto avviene durante la prima Bus Turnaround Phase.

3.3 Simulazione presa dati

Al fine di simulare il comportameto del firmware durante una presa dati sono stati ipotizzati due scenari di operatività. Il primo prevede di simulare una sorgente, posta quasi a contatto con il chip, che emette radiazione in modo uniforme su tutta la superficie della matrice dei pixel con una frequenza pari a 800 kHz. Questo scenario è poco fedele alle condizioni di lavoro che vengono richieste nell'ambito del progetto, però è di grande utilità per testare il comportamento del firmware a regimi elevati. Il secondo scenario, invece, è più fedele alle reali condizioni di lavoro e infatti si è simulata una sorgente puntiforme, posta a 1 mm di distanza dal chip, che emette con una frequenza di 10 kHz. In entrambi gli scenari, gli intervalli temporali tra una hit e la successiva sono stati simulati generando numeri casuali appartenenti a una distribuzione esponenziale $f(t) = \frac{1}{\tau}e^{-\frac{t}{\tau}}$, con il parametro τ espresso in unità di 100 ps, dal momento che l'interfaccia di simulazione di Vivado non consente di operare con unità di tempo arbitrarie. Inoltre, nel secondo scenario, la sorgente puntiforme posta a 1 mm di distanza dal chip è stata approssimata con una gaussiana in due dimensioni con valore medio $\bar{\mathbf{x}}$ pari alla coordinata del centro della matrice di pixel, $\bar{\mathbf{x}} = (512\,\text{px}, 256\,\text{px})$ e $\sigma = 0.5\,\text{mm}$ che corrisponde, viste le dimensioni medie di un pixel, a $\sigma = 18\,\text{px}$.

Nella figura 3.6 sono riportate le forme dei segnali più significativi durante una simulazione di presa dati nel caso di sorgente uniforme (nel caso della sorgente puntiforme non vi è alcuna differenza): i tre segnali in azzurro corrispondono alle coordinate dei pixel, generate casualmente, che vengono accesi, i due segnali in fucsia rappresentano rispettivamente l'output DDR a 4 bit e il segnale `strobe` del modulo `ALPIDE readout`, mentre i restanti segnali in arancio corrispondono rispettivamente all'output SDR a 8 bit e al segnale `valid_data` del modulo `readout deserializer`.

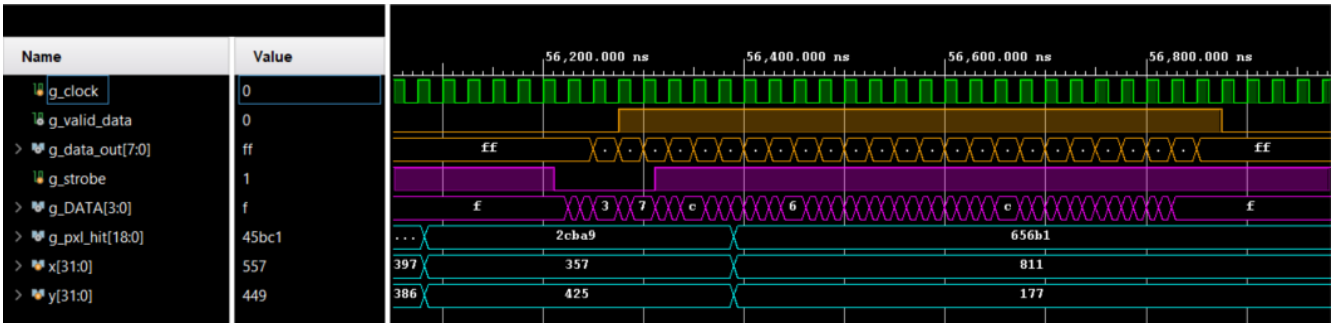

Figura 3.6: Forme dei segnali durante la simulazione di presa dati.

Sebbene la verifica dell'accuratezza della simulazione non può prescindere dal confronto delle forme dei segnali osservate con quelle attese, è possibile avere un riscontro più diretto

andando a salvare in un file di testo i valori assunti dal segnale `g_data_out[7:0]` in modo da ricostruire, via software, il numero di hit per ogni framing window, con le corrispettive coordinate dei pixel accesi. Ciò consente di realizzare un grafico cumulativo in cui viene mostrato il numero di hit registrate da ogni pixel per tutta la durata della simulazione effettuata. I grafici ottenuti nel caso di sorgente uniforme e nel caso di sorgente puntiforme sono riportati rispettivamente nelle figure 3.7 e 3.8, in cui i pixel accesi sono stati ingranditi per una migliore visibilità.

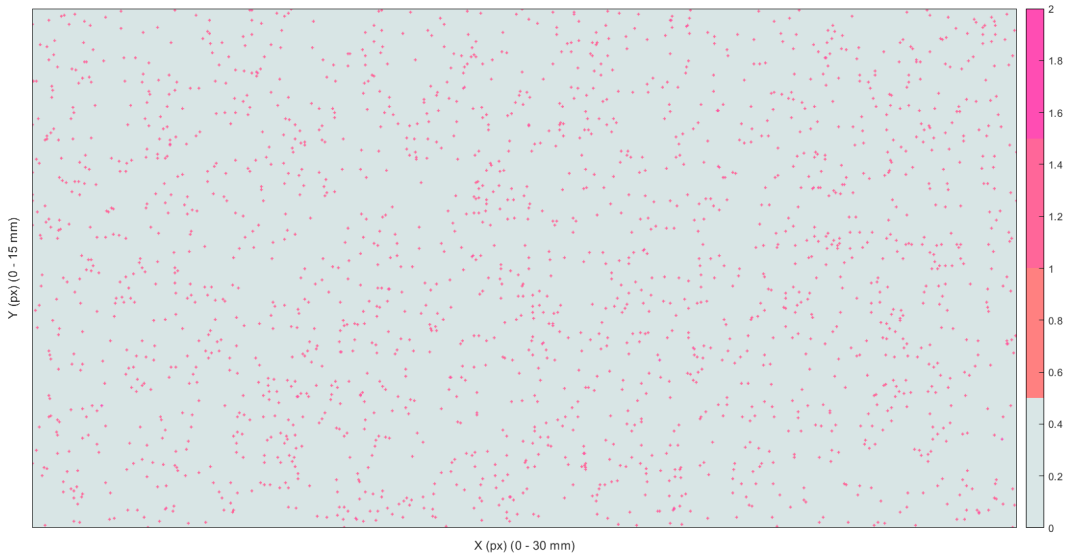

Figura 3.7: Grafico cumulativo delle hit registrate durante la simulazione con sorgente uniforme.

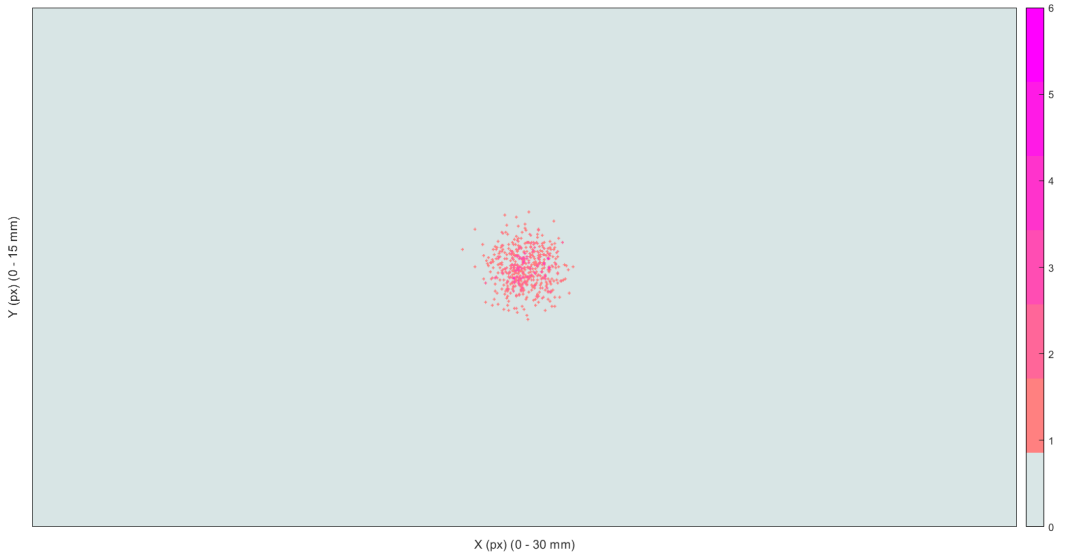

Figura 3.8: Grafico cumulativo delle hit registrate durante la simulazione con sorgente puntiforme.

Capitolo 4

Conclusioni

Le simulazioni effettuate e riportate nel Capitolo 3 hanno mostrato che il firmware sviluppato per l'FPGA soddisfa le funzionalità richieste riguardo sia la comunicazione che la trasmissione dei dati con ALPIDE. In particolare si è visto che per quanto riguarda la trasmissione dei dati, la velocità della porta parallela in DDR DATA[3:0], pari a 320 Mb/s è ampiamente sufficiente per le condizioni di operatività richieste al chip. Infatti il chip per trasmettere l'informazione relativa all'accensione di un singolo pixel impiega 48 bit (confronta la tabella 1.2) e quindi nel caso di una sorgente che emette con una frequenza di 10 kHz saranno necessari solamente 480 kb/s contro i 320 Mb/s disponibili. Seppure questa stima sia semplicistica, è facile notare che anche nel caso più realistico in cui una radiazione causi l'accensione di più pixel e considerando sempre sorgenti che emettono con una frequenza dell'ordine dei pochi kHz, la velocità di trasmissione di ALPIDE è più che sufficiente. Specifichiamo inoltre che queste considerazioni non tengono conto del tempo di drift delle cariche nel volume sensibile dei pixel, dello shaping time della catena elettronica presente all'interno di ognuno di essi e del tempo che impiega l'architettura digitale a portare il segnale dalla matrice alla periferia, fattori che richiedono uno studio più cospicuo.

Tra i possibili sviluppi futuri del lavoro svolto, oltre che la verifica del funzionamento del firmware realizzato in un caso reale con il chip ALPIDE operativo, rientra sicuramente il raffinamento dell'inizializzazione del chip tramite l'introduzione di algoritmi di *masking* e *pulsing*, che hanno rispettivamente il compito di disattivare degli eventuali pixel rotti o difettosi e iniettare della carica di prova all'interno dei pixel funzionanti in modo da verificarne l'integrità ogni volta che ALPIDE viene avviato. Per quanto riguarda il lato software, invece, risulterebbe particolarmente adatto all'applicazione pratica considerata un sistema di acquisizione in tempo reale dotato di GUI.

Bibliografia

[1] M. Mager, ALICE collaboration et al. “ALPIDE, the Monolithic Active Pixel Sensor for the ALICE ITS upgrade”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 824 (2016), pp. 434–438.

[2] Chris Damerell. *Tracking the rise of pixel detectors*. 2021. URL: https://cerncourier.com/a/tracking-the-rise-of-pixel-detectors/.

[3] ALICE ITS ALPIDE development team. *ALPIDE Operations Manual*. 2016.

[4] Stefania Beolé Luciano Musa. *ALICE tracks new territory*. 2021. URL: https://cerncourier.com/a/alice-tracks-new-territory/.

[5] IEEE Computer Society. *IEEE Standard for Verilog Hardware Description Language (IEEE Std 1364 - 2005)*. 2006.

[6] Inc Xilinx. *Artix-7 35T Arty FPGA Evaluation Kit*. URL: https://www.xilinx.com/products/boards-and-kits/arty.html#information.

[7] Inc Xilinx. *Xilinx Vivado*. URL: https://www.xilinx.com/products/design-tools/vivado.html.

[8] Inc Xilinx. *7 Series FPGAs SelectIO Resources User Guide*. 2018.