POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Elettronica

PROGETTO E SVILUPPO DI UN SISTEMA A 128 CANALI AD ALTO THROUGHPUT PER APPLICAZIONI FAST CORRELATION SPECTROSCOPY

Relatore: Prof. Ivan RECH
Correlatore: Ing. Ivan LABANCA

Tesi di Laurea di:

Federico RIGHETTI Matr. n. 787534
Pietro TROI Matr. n. 782852

Anno Accademico 2013-2014

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

Negli ultimi anni molte ricerche in ambito chimico e biologico si sono basate su sistemi in grado di compiere misure a singolo fotone, poiché questi sono in grado di rivelare bassissime intensitá luminose e consentono l'analisi di segnali con evoluzioni temporali ultra-brevi. Questi sistemi misurano la fuorescenza, ovvero il fenomeno di emissione di fotoni da parte di un campione sottoposto ad eccitazione con luce nello spettro del visibile.

Fra le varie applicazioni che sfruttano sistemi a singolo fotone ve ne sono due sulle quali é stato basato il progetto di tesi: la Fluorescence Correlation Spectroscopy (FCS) e la Förster Resonance Energy Transfer (FRET). La prima effettua la correlazione fra i vari canali di acquisizione dati per ricavare informazioni sulle cellule, mentre la seconda utilizza la tecnica di time stamping che va a registrare l'istante di arrivo dei fotoni rispetto all'inizio dell'esperimento.

Parallelizzando misure fatte con singoli sensori si ha un rapido incremento del throughput di dati da gestire (diversi Gb/s). Abbiamo quindi progettato e realizzato una scheda a 128 canali. Scopo di questa scheda é quello di soddisfare entrambe le richieste in termini di elevato numero di canali da gestire e velocitá di trafserimento.

Abstract

In recent years many researches in the biological and chemical fields were based on systems able to perform measurements using single photon device that can detect very low light intensity as well as being able to analyze signals with tight temporal evolution. These systems measure the fluorescence that represents photon emission from a sample being exited by light in the visible spectrum. Among those applications that rely on single photon detectors we focused our attention on two of them: Fluorescence Correlation Spectroscopy (FCS) and Förster Resonance Energy Transfer (FRET). The first one performs the correlation between various channels in order to aquire informations on the sample cells under test, while the second one exploits the time-stamping technique in order to record the arrival time of photons with respect of the beginning of the experiment. The substantial parallelism required by these applications entail for a dramatic increase in the throughput (several Gb/s). A 128-channels board was therefore designed. Aim of this board is to satisfy both high number of channels and substantial speed requests.

Introduzione

Oggigiorno molte applicazioni, specialmente nel campo della ricerca biologica si basano sull'analisi ottica. Recenti sviluppi nel campo della spettroscopia richiedono un incremento sempre maggiore in termini di sensibilitá dei foto-rivelatori in aggiunta ad un maggior grado di parallelizzazione degli esperimenti. La combinazione di questi due elementi comporta la ricerca

di una velocitá di trasferimento dati sempre crescente. Esempi di queste applicazioni sono rappresentate dalla smFRET (acronimo di single-molecule Förster Resonance Energy Transfer) e dalla FCS (acronimo di Fluorescence Correlation Spectroscopy).

La FRET rappresenta una interazione dipolo-dipolo tra due molecole fluorescenti. Questo meccanismo é ampiamente usato in spettroscopia per misurare la distanza tra due molecole fluorescenti dell'ordine dei 1nm-10nm, distanze queste che risulterebbe inaccessibile ad un microscopio convenzionale. Tale sensitivita é richiesta in ambito biologico e biofisico per studiare problemi relativi alla struttura delle biomolecole, la loro conformazione e la loro dinamica. Il criterio migliore per affrontare questi problemi é quello di osservare ad una singola molecola. La bassa concentrazione del campione in esame richiede lunghi tempi di misura in modo da poter raccogliere dati statisticamente rilevanti al fine di poter studiare lo stato di equilibrio delle biomolecole. La dinamica rimane invece nascosta nei primi secondi di misura. Una possibilitá per superare questa limitazione é rappresentata dall'utilizzo della single molecule FRET (smFRET) in cui la luce viene raccolta da piú sensori contemporaneamente permettendo di abbattere il tempo di misura di un fattore pari al numero di sensori utilizzato. Concentrazioni inferiori a 100pM sono necessarie per studiare singole molecole ma questo richiede

sensori altamente efficienti per catturare la fluorescenza emessa. Oggigiorno sensori a singolo fotone rappresentano la scelta piú indicata rispetto ai Photo Multiplier Tubes (PMT) per motivi di maggiore integrabilitá, robustezza e insensitivitá a disturbi elettromagnetici.

La FCS é una tecnica in grado di misurare fluttuazioni spontanee di molecole fluorescenti in volumi dell'ordine del femtolitro. La dipendenza temporale dell'intensitá della fluorescenza é analizzata in termini della sua funzione di autocorrelazione ricavando informazioni relative alla concentrazione e cinetica della molecole in esame. Recenti sviluppi in termini di fotorivelatori hanno portato ad una sempre maggiore parallelizzazione e conseguentemente un aumento del throughput di dati da analizzare.

Allo stato attuale é presente un sistema a 48 canali che effettua misure smFRET di durata massima di 167ms. Il nostro progetto ha come scopo quello di realizzare un sistema che raccolga dati da due array di sensori SPAD da 64 canali ciascuno.

Le specifiche richieste prevedevano un conteggio medio per ognuno dei 128 canali pari a 100kcps. Per ogni canale si richiede la trasmissione di 8 byte rendendo il throughput totale richiesto pari a circa 102MB/s totalmente insostenibile dal precedente protocollo USB2.0. Abbiamo quindi impiegato un protocollo USB3.0 gestito da un microcontrollore Cypress EZ-USB FX3 affiancato ad una FPGA che si occupa dell'elaborazione dati provenienti dai sensori. L'utilizzo di 8 byte, di cui 6 riservati al tempo di misura, permette di aumentare notevolmente il fondo scala dell'esperimento permettendo di eseguire misure di durata pari a circa 32 giorni.

Questo lavoro verrá ora descritto secondo la seguente struttura: Nel capitolo 1 verranno presentate due applicazioni per le quali e stato sviluppato il nostro lavoro di tesi.

Nel capitolo 2 verranno descritte le schede realizzate motivando le scelte fatte sia in termini di componenti che di layout.

Nel capitolo 3 verrá trattata la parte firmware e software che permette al sistema di comunicare da e verso verso PC.

Nel capitolo 4 verranno riportati i risultati dei test svolti per verificare il corretto funzionamento di quanto sviluppato nel capitolo 3.

Capitolo 1

Principali applicazioni

Questo capitolo introduce alcune definizioni chiave relative alla fluorescence correlation spectroscopy (FCS) cosi come la criticitá degli strumenti coinvolti per effettuarla. Verrá descritto il setup tipico con cui si svolgono esperimenti di FCS, il principio base di funzionamento di un rivelatore SPAD e il sistema tipico in cui si trova ad operare.

1.1 Fluorescence Correlation Spettroscopy (FCS)

La fluorescenza é l'emissione di luminescenza che si osserva a seguito dell'assorbimento di fotoni. La si distingue dalla sua corrispondente, la fosforescenza, la quale ha un tempo di decadimento maggiore.

La fluorescenza é la proprietá di alcuni atomi e molecole di assorbire luce a particolari lunghezze d'onda e successivamente di emettere luce a lunghezze d'onda maggiori, per un breve intervallo. Tale intervallo lo si puó quantificare come il tempo di vita della fluoresacenza ed é dell' ordine dei 10 ns invece il tempo di decadimento della fosforescenza copre un range che va dai millisecondi ai secondi.

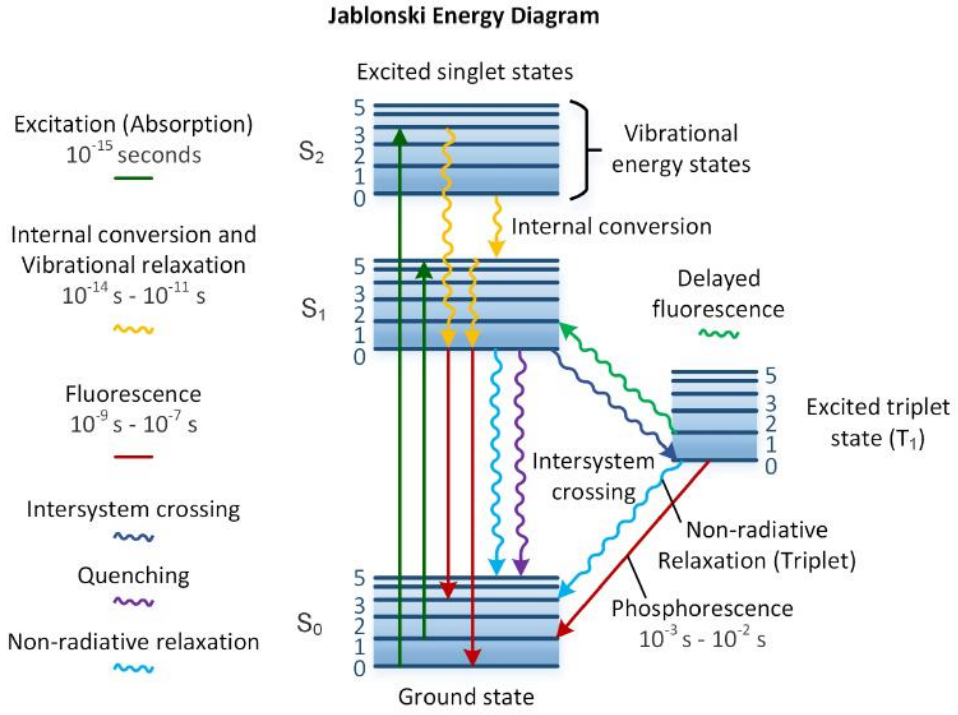

Figura 1.1: Rappresentazione del processo fondamentale della fluorescenza.

Come si vede in figura 1.1, il processo di fluorescenza puó essere suddiviso in tre importanti eventi dominati da grandezze temporali diverse:

- eccitazione della molecola a seguito dell'assorbimento di un fotone e quindi passaggio dell'elettrone piú esterno al livello energetico eccitato $(10^{-15}s)$;
- decadimento dell'elettrone dallo stato eccitato a quello ad energia inferiore a seguito di relazioni vibrazionionali nella molecola $(10^{-12}s)$;
- emissione di un fotone a lunghezza d'onda maggiore e ritorno dell'elettrone allo stato energetico base $(10^{-9}s)$.

Quindi l'intero processo, dall'assorbimento all'emissione, impiega qualche milliardesimo di secondo e rappresenta molto bene l'interazione tra la materia e la luce.

La FCS misura l'intensitá della fluorescnza spontanea generata da una singola molecola o da piú molecole in un volume di materia di circa $10^{-15}L$, illuminato da un raggio laser [1][2][3]. Il punto di forza di questa tecnica é che a differenza della single-molecule detection (SMD), non richiede che l'oggetto della misura sia statico, ma puó essere utilizzata pure per le molecole in una soluzione, infatti in questo caso, in seguito a fenomeni di diffusione nel volumetto, le molecole sono continuamente rimpiazzate. Quindi la FCS permette di poter eseguire misure continue per lunghi periodi, senza la necessitá di utilizzare molecole selezionate ed inoltre l'intensitá delle fluttuazioni é il risultato di alcuni processi dinamici, tipicamente della diffusione verso il volumetto e verso l'esterno di esso.

Quando i fluorofuori vengono colpiti dal raggio laser emettono un'insieme di fotoni a causa dei multipli cicli di eccitazione-emissione delle stesse molecole. L'analisi eseguita con la correlazione delle fluttuazioni della fluorescenza permette di poter ricavare le concentrazioni locali, coefficineti di diffusione, costanti chimiche, costanti di associzione e di dissociazione e le dinamiche strutturali.

Recentemente la FCS é stata utilizzata per ricavare il movimento delle molecole e la densitá dei recettori nelle cellule vive e nei tessuti [4][5][6][7][8]. Nonostante la tecnica sia stata utilizzata per monitorare molti processi chimici e biochimici, essa ha alcune limitazioni [9][10], ma per superare questi problemi, sono state sviluppate nuove tecniche come estensioni della FCS , le quali sono piú utilizzate nella misura del moto e della diffusione delle molecole nelle membrane. La *Fluorescence cross-correlation spectroscopy* (FCCS) é stata utilizzata per lo studio della dinamica e delle interazioni di molecole [11][12]. In aggiunta all' auto-correlazione e alla cross-correlazione vi é la *fluorescence lifetime correlation spectroscopy* (FLCS) che é la combinazione della FCS con la *Time-Correlated Single Photon Counting* (TCSPC), la quale rende possibile riconoscere il contributo dei fluorofori in un sistema di misura [13].

1.1.1 Principio base della FCS

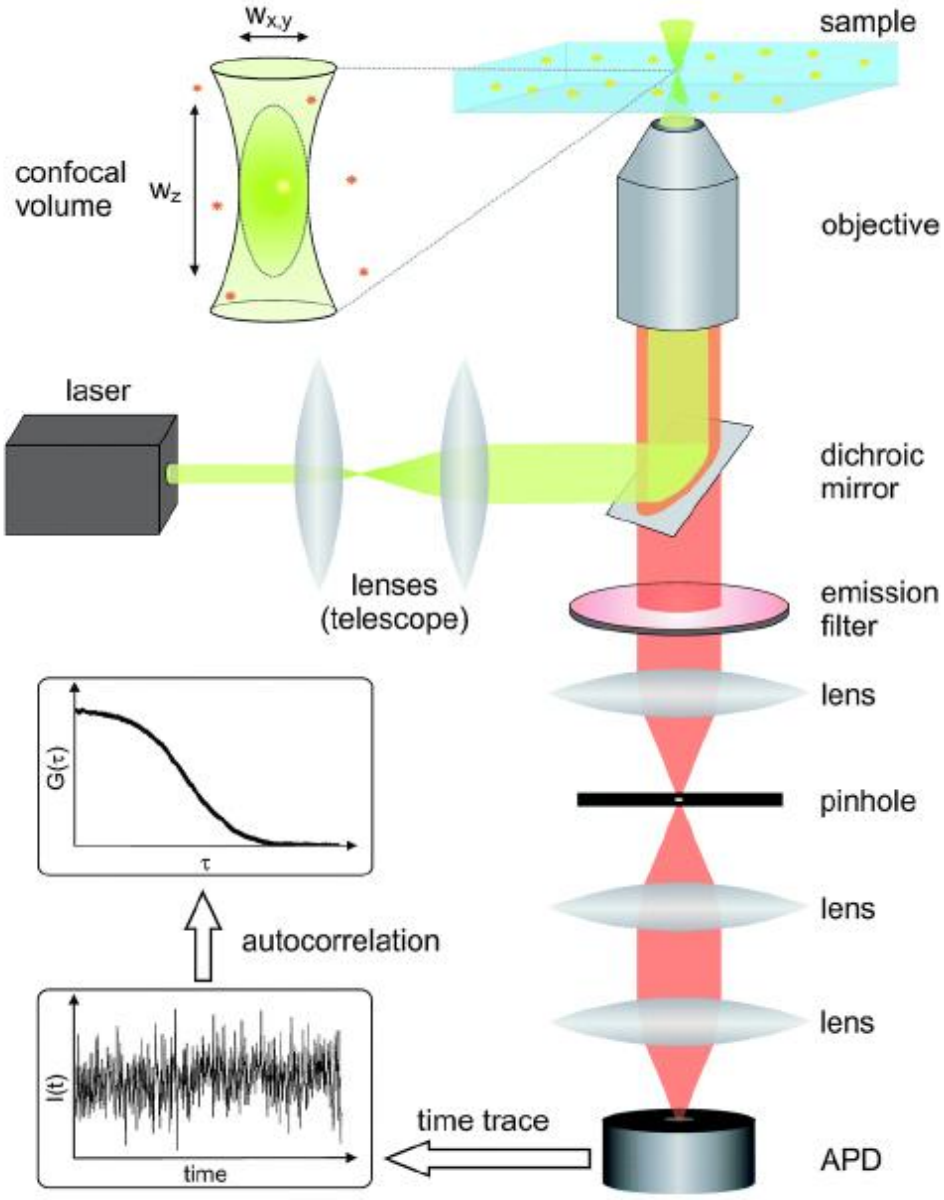

Figura 1.2: Setup tipico per esperimenti FCS.

In figura 1.2 é mostrato un tipico setup per esperimenti FCS [14]: si puó notare una lente confocale che viene utilizzata per rilevare la fluorescenza che viene emessa dalle molecole che passano attraverso un volumetto di qualche

femtolitro. A tale scopo, il volume viene creato focalizzando un laser dove si trova l'oggetto sotto misurazione e conseguentemente la fluorescenza emessa viene catturata dall'obiettivo, poi passa per il filtro di emissione, il quale evita che la luce che é stata emessa dal laser passi direttamente da lí senza passare dal volumetto. Successivamente passa per una lente che focalizza il fascio luminoso per fa si che possa passare tutto attraverso il pinhole (forellino). Alla fine é posizionato un detector molto sensibile, come ad esempio un avalanche foto diodes (APD). Questi sensori permettono di rilevare fluttuazioni dell' intensitá della fluorescenza che variano nel tempo e di salvarle come I(t). Successivamente il dato viene inviato ad un correlatore, il quale fornisce una stima della funzione di autocorrelazione su un certo range dinamico.

In particolare l'autocorrelazione (ACF) utilizza i dati salvati per eseguire tale funzione e ci fornisce informazioni sul comportamento medio di tutte le molecole sotto osservazione, analizzando il comportamento casuale delle singole. Per il calcolo della ACF, si va a comparare il dato misurato, con se stesso traslato nel tempo del τ ovvero del *lag time*. Nel caso in cui non ci siano traslazioni nel tempo, entrambe le tracce di dati sono identiche e la correlazione é alta invece se la traslazione é grande, le due tracce sono molto differenti e la correlazione é bassa.

Matematicamente l'ACF é rappresentata dalla formula (1.1):

$$\widehat{g}(\tau) = \frac{\langle I(t) \cdot I(t+\tau) \rangle}{\langle I(t) \rangle^2} - 1 \tag{1.1}$$

dove I(t) é il segnale intensitá al tempo t e $\langle I(t) \rangle$ é il tempo medio del segnale ed é espresso dalla formula (1.2):

$$\langle I(t) \rangle = \lim_{(T \to \infty)} \frac{1}{T} \int_0^T I(t) dt \tag{1.2}$$

Il decadimento che si puó osservare nel grafico della ACF in figura 1.2 ci sta ad indicare che, dove ho i valori maggiori si tiene conto del fatto che ogni molecola che fornisce un segnale di correlazione occupa il volumetto di osservazione per una piccola durata, mentre poche molecole transitano

nel volumetto per un piccolo periodo fornendo il decadimento che si vede nel grafico. Quindi si puó dire che l'ampiezza della ACF é inversamente proporzionale al numero medio di molecole presenti nel volumetto e la sua semilarghezza fornisce il tempo medio del suo attraversamento da parte delle molecole. Grazie a tali informazioni é possibile ricavare la dimensione del volumetto di test, la concentrazione e altri parametri che possono derivare dall'osservazione delle molecole.

1.1.2 Multispot FCS

La single-spot FCS trattata fino ad adesso, non é efficente poiché permette di utilizzare un solo laser spot e impiega qualche secondo per fornire il risultato. Per poter essere piú efficenti si passa alla parallelizzazione della FCS e ció permette di ottenere una piú rapida acquisizione dei dati e ció si puó rivelare molto vantaggioso come ad esempio quando si vanno ad esaminare le molecole di un reagente ed essendo distribuite nella soluzione, vi é la necessitá di un'analisi simultanea delle varie molecole. Per poter attuare una multisopt FCS, per il setup, per la parte di eccitazione, vi sono varie soluzioni: una soluzione per poter generare piú spot, riguarda l'utilizzo di un beam splitter [15] per laser a stato solido [16], il tutto arraggiato in una configurazione con array 8x8. Un altro metodo utilizza cristalli liquidi su silicio [17] che permettono di generare un array di 32x32 spot [18].

Passando alla parte di ricezione, dobbiamo utilizzare dispositivi con aree attive multiple. Esistono in commercio sistemi di ricezione 8x8 basati su PMTs (photo-multiplier tubes) [19] e SPCMs (solid-state single-photon-counting modules). Purtroppo, attualmente gli SPCMs non possono essere integrati in array ad alta densitá [20], poiché la loro struttura ha bisogno di un sistema di raffreddamento e di un circuito di quencing, inoltre l'elettronica di uscita non puó essere integrata sullo stesso chip, poiché nel processo di produzione viene utilizzato un processo non compatibile con i CMOS. Molto utilizzati sono invece i rivelatori a singolo fotone, *single photon avalanche diode array* (SPAD arrays), i quali verranno descritti nel paragrafo 1.3.4 insieme al sistema all'interno del quale lavorano. La multispot FCS ci per-

mette di introdurre il *cross-correlatore digitale* , infatti a seguito dei molti dati generati dai rilevatori di fotoni multi-pixel, sono stati sviluppati vari correlatori hardaware che permettono di ottenere risultati in tempo reale e con un ampio range di lag time. A livello software sono stati sviluppati vari algoritmi ottimizzati per poter perfezionare ulteriormente i risultati della FCS ed infine vi sono correlatori multicanale basati su FPGA che permettono di ottenere buoni risultati.

1.1.3 Cross-correlatori digitali

I *cross-correlatori digitali* (CCF) sono espressi tramite una formula (1.3) molto simile a quella dell' ACF (1.1):

$$\widehat{g_{xy}}(\tau) = \frac{\langle I_x(t) \cdot I_y(t+\tau) \rangle}{\langle I_x(t) \rangle \langle I_y(t) \rangle} - 1 \tag{1.3}$$

dove $I_x(t)$ e $I_y(t)$ sono le intensitá delle fluttuazioni dei due segnali che devono essere correlati. Come si puó osservare dalle formule (1.1) e (1.3) le due differiscono solo per gli ingressi, infatti nel caso dell'autocorrelatore i due ingressi del cross-correlatore sono sostituiti con due uguali ingressi, per cui da adesso parleremo solo di *correlatore*.

La funzione di correlazione é stata inizialmente sviluppata seguendo uno schema lineare, il quale ha uno stesso tempo di campionamento per tutti i canali. Poi si é passati al *correlatore esponenziale* e successivamente al *correlatore multi-tau* il quale ha permesso di poter aumentare il range del lag time [21] rispetto all'algoritmo precedente, utilizzando diversi lag time seguendo una scala logaritmica. Prima di trattare i 3 correlatori elencati bisogna tener conto di quattro importanti parametri :

- Δt tempo di campionamento che é l'inverso del clock di sistema ;
- τ lag time;
- T_c tempo di coerenza;
- T tempo totale dell'esperimento.

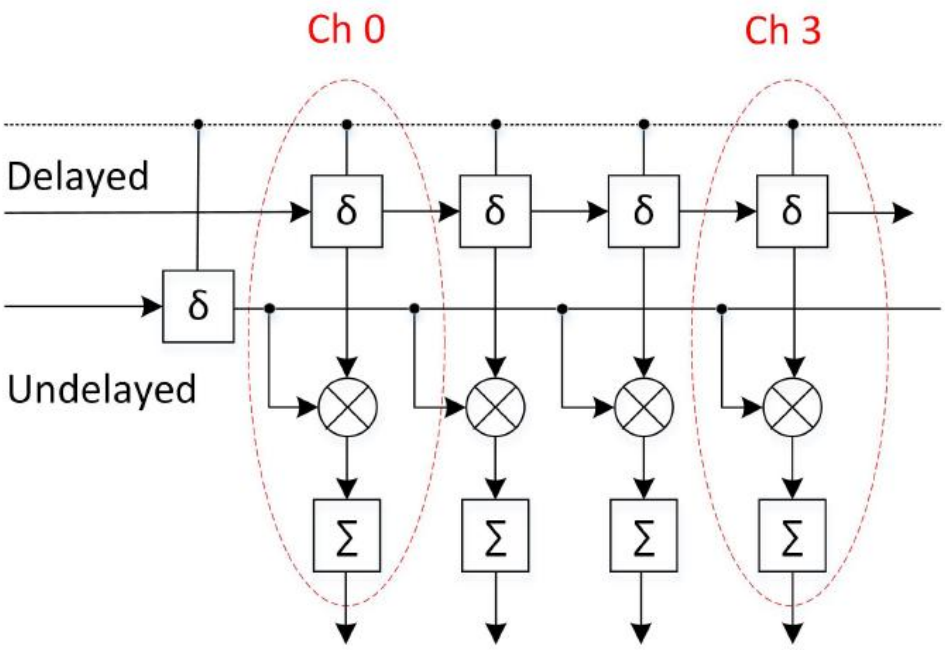

Figura 1.3: Struttura di correlatore lineare: il canale Ch0 ha lag time zero, mentre Ch3 ha lag time pari a $3 \cdot \Delta t$

La relazione che si ha tra questi quattro é $\Delta t \ll T_c \ll T$ mentre $\tau_{max} \geq T_c$ e $\tau_{min} \geq \Delta t$ con τ_{max} e τ_{min} che sono rispettivamente il massimo e il minimo del lag time.

1.1.4 Correlatore lineare

Come si puó vedere dalla figura 1.3 il *correlatore lineare* é costituito da piú canali, ognuno dei quali permette di fare moltiplicazione e accumulazione e inoltre svolge la correlazione per un certo lag time simultaneamente e indipendentemente. I differenti lag times sono distanziati di un intervallo di campionamento da quello precedente come si puó vedere dallo schema e il numero totale di lag channels é pari a $\tau_{max}/\Delta t$.

Questo non é un modo molto efficente per effettuare la correlazione, infatti se il tempo di campionamento dei fotoni per esempio é di 10 ns, mentre il

lag time ha un range che va da 10ns a 100ms, il tutto risulta in circa 10^7 possibili lag time channels e ció implica un enorme consumo di risorse.

1.1.5 Correlatore esponenziale

Dopo il correlatore lineare, é stato sviluppato il *correlatore esponenziale*, nel quale i ritardi dei canali successivi vengono moltiplicati per un fattore costante 2^m. Questo permette di ottenre un ampio range dinamico, ma nonostante ció, per ampi lag time, il campionamento del correlatore risulta particolarmente grossolano, infatti all'aumentare del lag time, molte informazioni vengono perse.

1.1.6 Correlatore multi-tau

Per superare i problemi nati con i correlatori precedenti é stato sviluppato il *correlatore multi-tau* [22] il quale separa tutti i lag times nei vari blocchi ed ogni blocco ha un differente tempo di campionamento che é un multiplo del tempo di campionamento del blocco precedente. In poche parole, piú correlatori lineari costituiscono un correlatore multi-tau.

Il tempo di campionamento di ogni blocco puó essere espresso come:

$$\Delta t_i = \Delta t_o \cdot m^i; i = 0, 1, 2, ..., S-1 \tag{1.4}$$

dove Δt_0 é il tempo di campionamento del correlatore lineare piú veloce, il quale utilizza il minimo lag time, S é il numero totale di correlatori lineari e m é il multiplo.

Per quanto riguarda il lag time in ogni blocco, esso sará:

$$\tau_i(p) = \tau_i(0) + \Delta t \cdot p. \tag{1.5}$$

La maggior parte dei correlatori commerciali viene settato con: m = 2 e p=16.

Dallo schema in figura 1.4 si puó osservare che il primo blocco ha il tempo di campionameto iniziale Δt_0 e lag time con un range: $0 \sim 15\Delta t_0$; il secondo blocco contiene un altro correlatore lineare, il quale ha un doppio tempo di

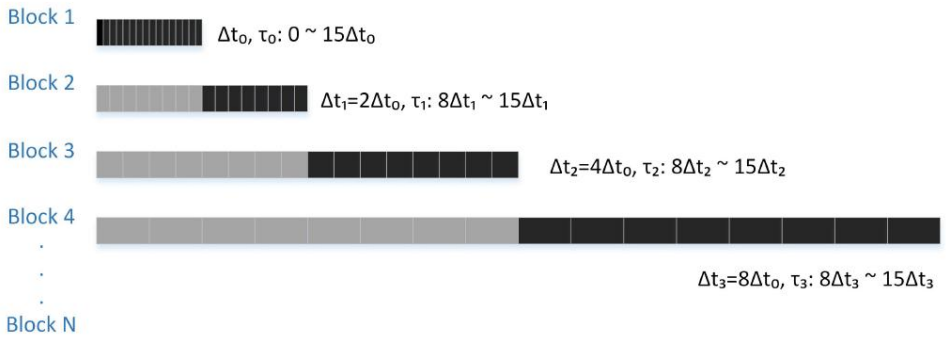

Figura 1.4: rappresentazione di un correlatore multi-tau: divisione in blocchi, divisione del tempo di campionamento e dei lag time.

campionamento rispetto al primo blocco e un lag time con un range effettivo: $8\Delta t_1 \sim 15\Delta t_1$, dal momento che i primi 8 lag time sono coperti dal primo blocco. Per i blocchi successivi si ripete quello appena descritto.

1.2 Stato dell' arte

1.2.1 Correlatori hardware

Correlatori di tipo hardware sono implementati soprattutto usando circuiti integrati appositamente progettati per lo scopo (ASIC). Il vantaggio di una tale implementazione risiede in una elevato grado di densitá di integrazione rispetto ad un dispositivo programmabile. Sono basati principalmente su algoritmi di tipo multi-tau in grado di coprire una dinamica che varia da pochi nanosecondi a diverse ore. Uno svantaggio dei correlatori di tipo hardware é dovuto al loro elevato costo e tempo di sviluppo. Per limitare questi aspetti negativi, la produzione di dispositivi ASIC differenti fra loro viene effettuata sfruttando lo stesso wafer di silicio. Questo approccio limita fortemente l adattabilitá dei dispositivi ASIC alle varie applicazioni.

1.2.2 Correlatori basati su dispositivi FPGA

Oggigiorno i dispositivi FPGA rappresentano una valida alternativa ai dispositivi ASIC in termini di costi e implementazione. Presentano l'innegabile vantaggio di non avere costi di sviluppo essendo dispositivi pronti all'uso e la flessibilitá che permette di essere riconfigurati con facilitá per adattarsi a scopi e soddisfare specifiche differenti. Correlatori basati su FPGA sono caratterizzati dall' avere quattro componenti fondamentali: un FPGA il quale viene programmato tramite un linguaggio VHDL o Verilog, un microcontrollore in grado di spedire i dati elaborati ad un PC, una PCB (printed circuit board) sulla quale viene montata l FPGA e il microcontrollore, e sia in grado di interfacciarsi ai rivelatori di fotoni, e infine un programma installato su un PC in grado di interfacciarsi (tipicamente si utilizza linguaggio C# o LabView) l FPGA, ricevere i dati elaborati e visualizzare il risultato della correlazione. Le prestazioni di tali tipi di crosscorrelatori dipendono dall ottimizzazione del codice VHDL e dal tipo di FPGA scelto.

1.2.3 Correlatori software

Questo tipo di correlatori presentano una flessibilitá ancora maggiore rispetto ai correlatori basati su FPGA in quanto il codice é direttamente implementato su un PC tramite programmi come C#. Per acquisire la sequenza di impulsi dai rivelatori, é richiesta una scheda di acquisizione apposta da collegare al PC. L'aquisizone puó avvenire in due modi differenti:

Time mode: viene misurato il numero di fotoni per ogni intervallo di tempo.
Photon mode: viene misurato il tempo tra l'arrivo di un fotone e l'altro

Il trasferimento dati puó essere effetuato direttamente.

L'elaborazione avviene usando uno schema multi-tau [23]. Poiché esiste un limite alla velocitá di trasferimento dati e alla dimensione del buffer, tipicamente i correlatori software hanno un limite superiore al conteggio di fotoni per evitare un overflow di conteggio se usati in modalitá Photon mode o di risoluzione se usati in modalitá Time mode . Inoltre questo tipo ci

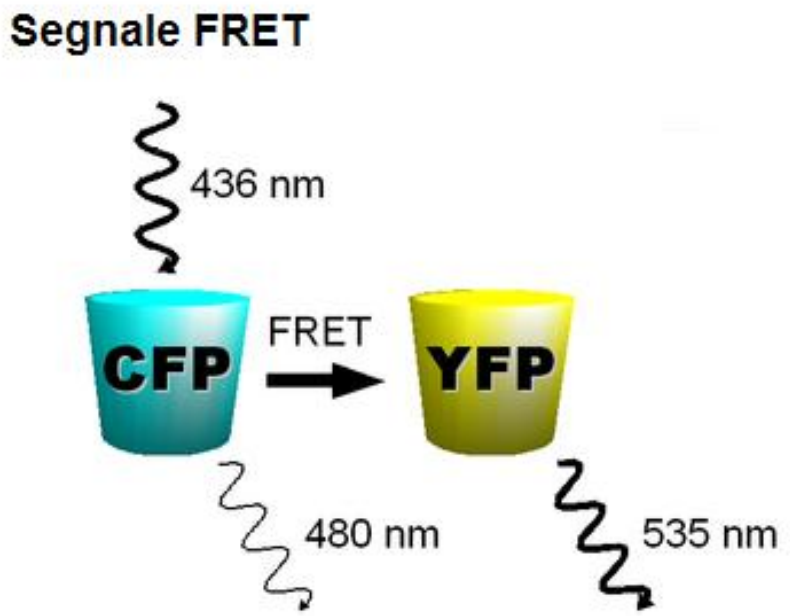

Figura 1.5: Schema della FRET: il donatore (CFP) viene eccitato da un'onda esterna; la molecola emette energia che puó essere trasmessa all'accettore (YFP). Se l'accettore si trova nelle vicinanze é possibile rilevare un segnale FRET.

cross-correlatori richiedono un elevao dispendio di risorse CPU rendendoli poco adatti per applicazioni real-time in esperimenti FCS, ma risultano piú adatti per analisi offline.

1.3 Single molecule FRET analisys

Il termine FRET (*Förster Resonance Energy Transfer*), sta ad indicare un'interazione non radioattiva del tipo dipolo-dipolo nella quale un fluorofo-ro, chiamato donatore, viene eccitato. Esso puó trasferire la sua energia a un secondo fluoroforo vicino a lui, chiamato accettore, portandolo ad emettere un fotone come mostrato in figura 1.5. L'efficineza di questo fenomeno la si puó osservare nella eq.(1.6):

$$E = \frac{1}{1 + (\frac{r}{R_0})^6} \tag{1.6}$$

dove con r viene indicata la distanza tra le molecole e con R_0 la distanza critica di Foster, un parametro legato alle particolari caratteristiche della coppia di molecole che fortunatamente oscilla solitamente tra i 2 ns e 6 ns, che é una distanza molto interessante per svolgere studi sulle macromolecole biologiche, sulla loro struttura, conformazione e dinamica [24][25][26]. A questo scopo, questa tecnica viene molto utilizzata in biologia, biofisica e biochimica [27][28][29]. In particolare noi tratteremo un'applicazione della FRET: la single molecule FRET (smFRET) perché andiamo ad osservare la singola molecola e poi ne calcoliamo l'efficienza. Questa tecnica permette un miglior studio dei campioni, perché analizzando la singola particella é possibile distinguere l'ammontare delle sub-popolazioni, che sarebbero impossibili da analizzare con gli strumenti di misurazione.

Livelli di concentrazione minori di 100 pM devono essere analizzati in questo regime, peró é necessario utilizzare detector con alta sensibilitá per rilevare la fluorerscenza emessa. Questi rilevatori possono essere Photo Multiplier Tubes (PMTs) o Single Photon Avalanche Diode (SPADs).

Gli SPAD, al giorno d'oggi, sono molto piú utilizzati dei PMT per la maggior intensitá del campo magnetico, minor dimensione, minor livello di alimentazione e la loro robustezza. In particolare, gli SPADs sviluppati al Politecnico di Milano, hanno un'alta efficienza che ha permesso di abbassare di molto il Dark Count Rate andando ad aumentare la sensibilitá sopra il background [30], ma nonostante tutto, la bassa concentrazione di campioni, richiede misure di alcuni minuti per poter raccogliere dati utili. Questo rende possibile solamente lo studio degli stati di equilibrio delle biomolecole, mentre la dinamica, rimane nascosta nei primi secondi di misurazione. Per superare questa limitazione é possibile utilizzare un setup multi-spot, dove la luce viene rilevata da piú di uno spot contemporaneamente e questo permette di abbattere il tempo di misurazione di un fattore pari al numero di spot [31][32][33].

Nel nostro caso andremo ad utilizzare 128 spot.

1.3.1 Setup per misurazioni single-spot smFRET

Questa tecnica, per concentrazioni di biomolecole inferiori a 100 pM, richiede l'utilizzo di un laser. Successivamente la luce riemessa dalle particelle, dipendente dall'efficinza della FRET, puó essere *rossa*, nel caso di un accettore perché si ha alta efficienza, oppure *verde* se si ha a che fare con il donore, perché si ha bassa efficienza.

Grazie all'utilizzo di uno specchio dicroico é possibile separare i due fasci luminosi per orientarli verso due sensori differenti e poiché le lunghezze d'onda in esame sono deboli, vi é la necessitá di utilizzare sensori in grado di rilevare i singoli fotoni. Inoltre ogni volta che il fotone viene rilevato, viene salvato il tempo di arrivo rispetto all'inizio del processo di misurazione. Una volta finita la misura e salvati i relativi tempi, é possibile inserirli in un grafico temporale della luce emessa: il fascio di luce emesso dalla molecola appare quando quest'ultima attraversa il fascio laser e viene rilevato solamente quando supera una determinata soglia. Per ogni fascio giunto al detector, ne viene calcolata l'efficienza basandosi sui fotoni rilevati da entrambi i sensori. Successivamente viene inserita in un'istogramma temporale e con ció é possibile studiare la distanza tra i fluorofori, che in ultima analisi rappresenta la struttura delle biomolecole.

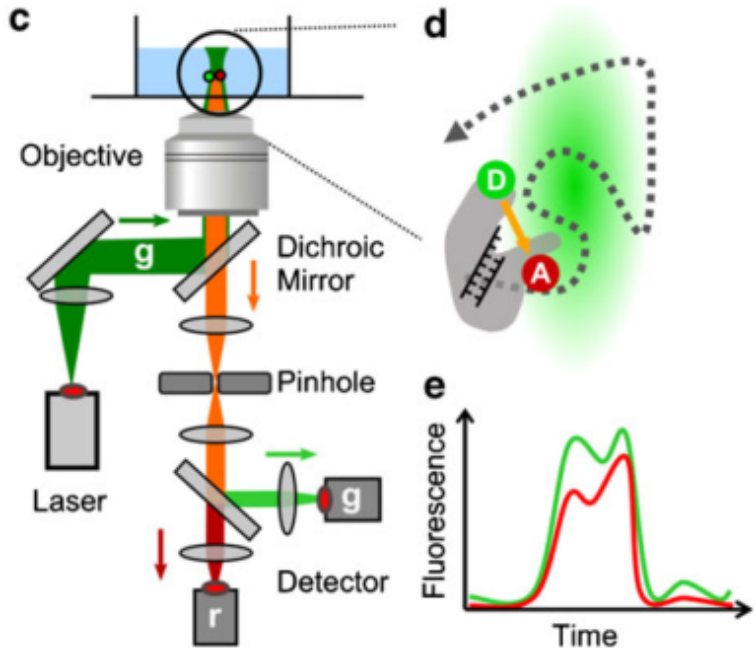

Figura 1.6: setup ottico per eseguire la Single spot smFRET. In particolare si possono osservare i due specchi microici, per separare i raggi luminosi,i due sensori spad, rispettivamente r e g e l'istogramma con efficineza e tempo.

Per quanto riguarda il setup optico si puó fare riferimento alla figura 1.6.

Il laser emette luce verde a 532 ns e viene indirizzato verso lo specchio dicroico. Successivamente, il fascio indirizzato verso le biomolecole, viene concentrato dalla lente del microscopio per generare un piccolissimo spot di luce ed andare a colpire con il suo fascio, l'oggetto della misura. Una volta colpite le biomolecole, esse riemettono la luce in modo isotropico, per cui solo una piccola frazione viene riemessa indietro verso la lente. Di questa frazione, solo una parte passa attraverso il filtro utilizzato per evitare che la luce del laser raggiunga i detector (che li potrebbe danneggiare). Infine vi é uno specchio dicroico che separa il fascio luminoso in luce verde e rossa e le indirizza verso i rispettivi detector. Da quanto detto si puó capire che l'efficienza dei componenti ottici é cruciale per la correttezza e l'efficacia del processo di misurazione.

1.3.2 Setup per misurazioni multi-spot smFRET

Per poter effettuare una smFRET multi-spot, figura 1.7, é necessario aggiungere altri componenti al setup iniziale, principalmente espansori del fascio laser e generatori di pattern. Poiché si vuole illuminare tutti gli spot con lo stesso fascio laser, bisogna generare un pattern che abbia questa funzione ed é possibile farlo grazie all'uso di LCOS (Liquid Crystal On Silicon). Questo é un disopsitivo, costituito da un cristallo liquido, che permette di modulare la fase della luce che lo colpisce, cambiando la polarizzazione di ogni suo pixel. Questo oggetto ha la tendenza a focalizzare la luce riflessa su un piano, per cui serve una lente per collimare gli spot che entrano nel microscopio.

Uno spot piú largo del fascio laser é necessario per poter regolare la risoluzione del LCOS con la dimensione dello spot del detector, per cui sono richiesti uno o piú espansori di fascio.

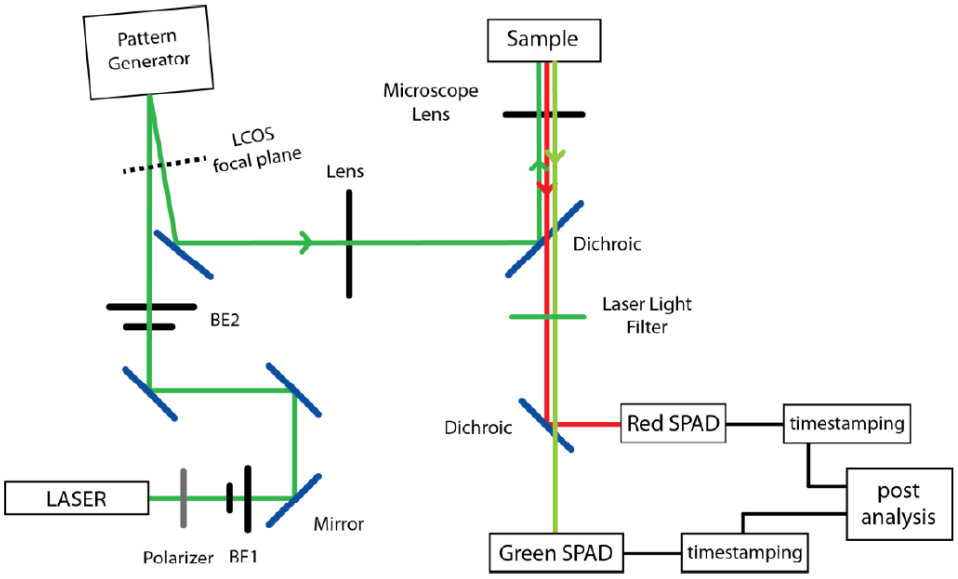

Figura 1.7: Quando viene effetuata una smFRET multi-spot, é necessario utilizzare un generatore di pattern. Esso lavora con luce polarizzata per cui bisogna introdurre all'inizio della catena un polarizzatore .

1.3.3 Allineamento

Il processo di allineamento consiste in un micro posizionamento dei detector in modo che, la luce che li colpisce, massimizzi il conteggio degli spad. Questa procedura risulta semplice con le misure single spot e invece molto piú complesso per quelle multi-spot poiché in questo caso il processo diventa iterativo e si divide in due step, uno automatico e l'altro meccanico. Per prima cosa, bisogna generare i vari spot con il LCOS e utilizzarli per testare una soluzione altamente concentrata (100 nM), di modo che la luce possa successivamente raggiungere i detector. La potenza del laser é regolata in modo che il rapporto segnale-rumore sugli SPAD sia alto abbastanza da rendere trascurabile il conteggio di buio, ma basso a sufficenza per evitare che lo SPAD oscilli. Adesso si puó cominciare con il primo passo dell' allineamento e si parte dal primo detector: il LCOS viene programmato per produrre un singolo spot indirizzato al primo SPAD e successivamente una procedura automatica scansiona le cordinate x e y in modo da trovare la posizione dove é stato rilevato il massimo segnale (massimo conteggio). Poi lo stesso viene ripetuto con il secondo detector creando un altro spot.

Successivamente, conoscendo le coordinate di entrambi gli spot, é possibile correggere la posizione e l'orientamento del primo detector.

Ripetendo il tutto in modo iterativo, é possibile fare in modo che si raggiunga il massimo su ogni SPAD di entrambi i detector e cosí si puó considerare concluso il processo di allineamento.

Bisogna peró tener presente che é molto facile alterare la stabilitá del sistema, infatti a seguito del posizionamento micrometrico, basta un piccolo movimento dei detector o di qualsiasi altro componente utilizzato nel setup per compromettere le misure. Inoltre il tavolo dove vengono svolte le misure deve essere sospeso per evitsare vibrazioni e infine si deve tener conto delle variazioni termiche dei componenti. Per esse risulta importante che non ci siano durante l'arco di tempo della misura, mentre invece non é importante se ci sono successivamente, perché una volta raccolti i dati, essi vengono normalizzati.

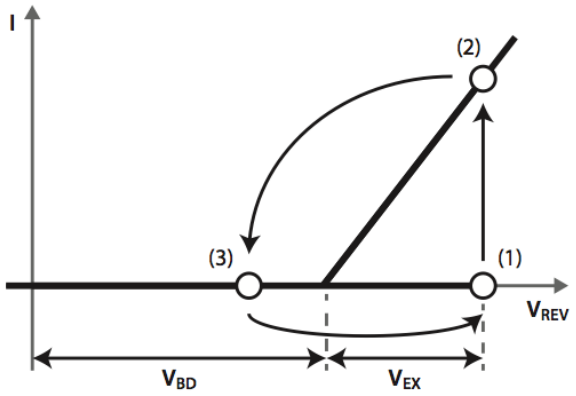

Figura 1.8: Grafico corrente tensione di uno SPAD.

1.4 Single Photon Avalanche Diode

Come accennato nei precedenti paragrafi, in applicazioni FCS e smFRET [34] si richiedono rivelatori ad alta efficienza. La tecnica del conteggio a singolo fotone rappresenta oggigiorno una valida alternativa rispetto ad una rivelazione analogica poiché si riescono a raggiungere maggiori risoluzioni temporali e minore sensitivitá ai rumori. Uno SPAD rappresenta una scelta vincente anche perché é possibile una sua integrazione come dispositivo a stato solido.

Lo SPAD é un dispositivo costituito da una giunzione p-n. Nello stato quiescente rappresentato da (1) in figura 1.8 lo SPAD é polarizzato inversamente con una tensione superiore alla sua tensione di breakdown ovvero:

$$V_{spad} = V_{ex} + V_{bd} \tag{1.7}$$

con V_{ex} tensione in eccesso rispetto alla tensione di breakdown (V_{bd})

Un fotone incidente lo SPAD genera una coppia di portatori (elettrone lacuna) che raggiunta la regione svuotata della giunzione si moltiplicano per un effetto di ionizzazione ad impatto e generano una corrente dell' ordine dei microampere rappresentata dal punto (2) in figura 1.8. Tale corrente genera una differenza di potenziale ai capi della resistenza R_B rappresentata in figura 1.9 che viene letta da un circuito di pick-up. A seguito di un fotone incidente lo SPAD non é piú in grado di reagire ad ulteriori fotoni incidenti motivo per

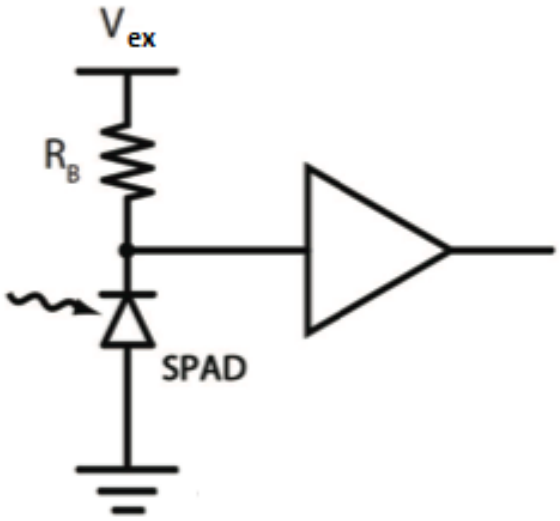

Figura 1.9: Figura rappresentante uno SPAD collegato ad una resistenza R_B che funge da circuito di quenching. Al catodo é collegato un circuito per il prelievo del segnale (pick-up)

cui si aggiunge un circuito di quenching che ha come scopo di interrompere la corrente di valanga, portando lo SPAD ad una tensione minore di quella di breakdown (3). Lo SPAD risulta di nuovo attivo riportandolo alla tensione $V_{bd} + V_{ex}$. La struttura cosí descritta si dice operare in modalitá Geiger. Uno schema circuitale di quanto descritto é rappresentato in figura 1.9.

Il vantaggio di una struttura cosí descritta é rappresentato dal segnale di uscita dello SPAD che risulta essere di tipo digitale con vantaggi dovuti ad una maggiore immunitá ai disturbi che porta ad avere un maggiore rapporto segnale/rumore dell' intera catena di acquisizione.

In un dispositivo SPAD sono comunque presenti fonti di rumore come risultato di valanghe non causate da fotoni incidenti. I due fenomeni che contribuiscono maggiormente a questi rumori sono:conteggi di buio e afterpulsing effect.

1.4.1 Conteggi di buio

Idealmente se nessun fotone colpisse lo SPAD non si attiverebbe nessuna corrente di valanga. Nella realtá coppie elettrone/lacuna possono essere generate

anche per via termica [35]. Numericamente si caratterizza questo fenomeno definendo un conteggio di buio (Dark Count Rate - DCR). Questo fenomeno puó essere limitato tramite una attenta ingegnerizzazione del dispositivo. In particolare i conteggi di buio sono dovuti a difetti e impuritá che possono generare una coppia elettrone/lacuna pur senza avere un fotone incidente. É importante quindi ridurre al minimo il numero di difetti e impuritá del dispositivo le quali risultano proporzionali all' area del dispositivo. Da un punto di vista operativo il DCR é funzione anche della temperatura e dal valore della tensione di polarizzazione dello SPAD. Curando il processo di produzione degli SPAD é possibile ridurre il fenomeno di tunneling che risulta essere dominante a basse temperature. Questa tecnica é stata usata al Politecnico di Milano permettendo di ottenere valori di DCR di 10 cps (conteggi per secondo) a -20^oC. Una maggiore tensione di polarizzazione dello SPAD comporta una maggiore efficienza del tunneling assistito da trappole che contribuisce al fenomeno fisico del conteggio di buio. Il valore medio del DCR puó essere misurato e sottratto mentre le sue fluttuazioni statistiche portano ad un rumore effettivo nel dispositivo. É possibile dimostrare che il numero di conteggi di buio in un certo intervallo temporale é una variabile casuale rappresentabile tramite una curva Poissoniana pertanto la varianza della variabile casuale é pari al suo valor medio.

1.4.2 Afterpulsing effect

Durante la valanga una corrente scorre nel dispositivo per un certo intervallo di tempo. Difetti presenti nella giunzione PN esiste una probabilitá che uno dei portatori che attraversino la giunzione sia intrappolato dal corrispondente livello di energia del difetto. Portatori intrappolati e successivamente rilasciati possono a loro volta innescare una valanga determinando un fenomeno di afterpulsing. In questo meccanismo di generazione si rumore é presente quindi una correlazione temporale con límpulso precedente. Diversi fattori contribuiscono al fenomeno dell' afterpulsing

- numero e tipo dei difetti presenti nel dispositivo é quindi importante curare il processo di fabbricazione dello SPAD.

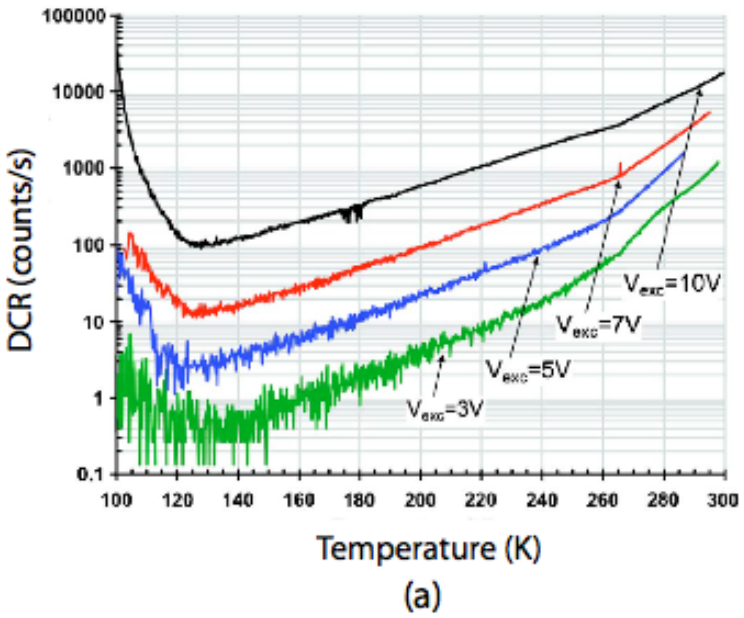

Figura 1.10: DCR di uno spad in funzione della temperatura per 3 differenti tensioni di polarizzazione.

- Numero di portatori che attraversano il dispositivo durante una valanga. É possibile limitarlo riducendo tramite appositi circuiti la corrente che scorre nel dispositivo.

Considerando i due fenomeni appena descritti possiamo evidenziare che per basse temperature domina il fenomeno di afterpulsing poiché aumentano i tempi di intrappolamento dei portatori. Nonostante il rumore diminuisca per basse temperature al di sotto dei 150K esso aumenta come mostrato in figura 1.10.

All' interno del nostro gruppo di ricerca, vengono usate matrici di SPAD all'interno di un dispositivo comunemente chimamato *manta*.

1.4.3 MANTA

La manta é un dispositivo composto da 3 parti principali rappresentati in figura 1.11:

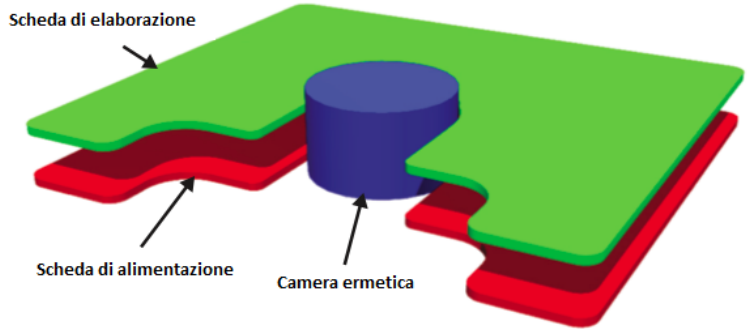

Figura 1.11: Disposizione delle tre parti principali che costituiscono una manta

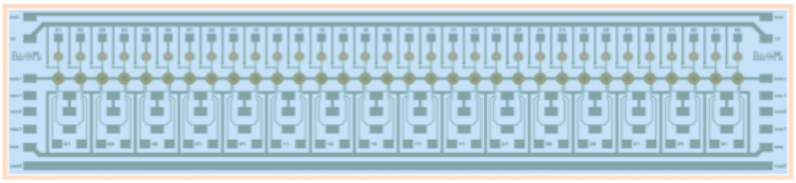

Figura 1.12: Layout del sistema da 32x1 SPAD.

Una camera ermetica contenente la matrice di SPAD, una scheda di elaborazione e una scheda di alimentazione. Questi elementi sono posizionati all' interno di un contenitore di alluminio di dimensioni compatte (135mm x 119mm x 26mm) e collegate tra loro in modo da poter fornire la corretta alimentazione a tutto il dispositivo. La scelta di dividere la parte di alimentazione da quella di lettura del segnale, é dettata dal voler ridurre i disturbi elettromagnetici iniettati nella scheda di acquisizione legati alla presenza di convertitori switching nella parte di alimentazione. Inoltre motivazioni termiche legate ad un migliore dissipazione di temperatura rendono favorevole questa scelta. Il nucleo della manta é costituito da una matrice di 32x1 SPAD, in figura 1.12 e figura 1.13 sono rappresentati il layout e la matrice rispettivamente.

La matrice é costituita da rivelatori con area attiva di 50um separati da un passo di 250um. Questa separazione é il risultato di un compromesso tra una distanza minima data dalla necessitá di routare gli SPAD e una massima data dal diametro dei vetrini attraverso cui filtra la luce per essere rivelata.

La tecnologia con cui il nostro gruppo di ricerca ha realizzato gli SPAD

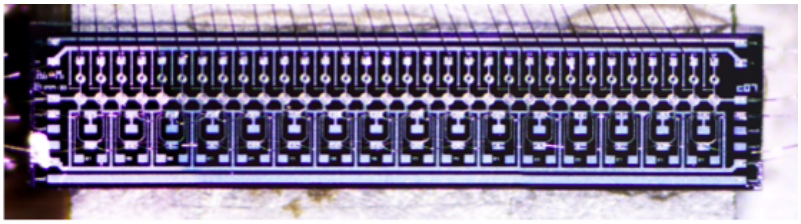

Figura 1.13: Immagine dell' array da 32x1 SPAD. Sono visibili i bonding.

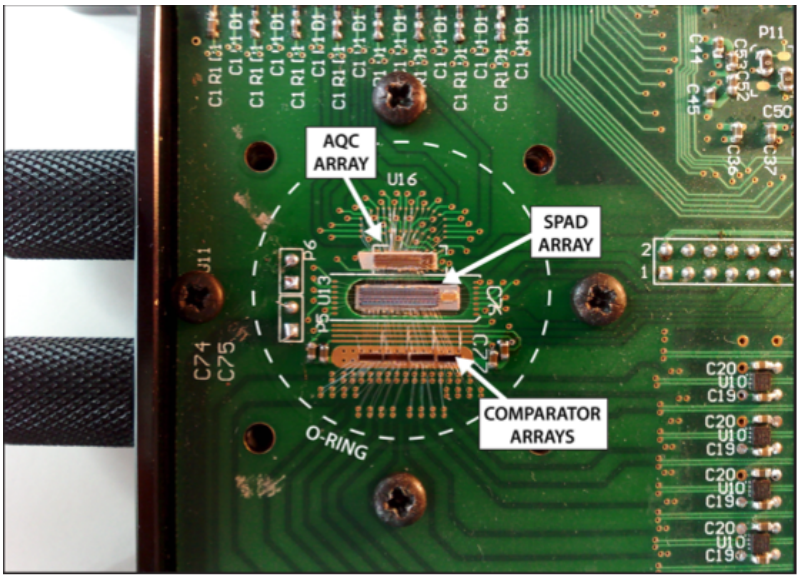

Figura 1.14: Disposizione dell'array all'interno della camera ermetica. Sono visibili i fili di bonding che collegano il circuito integrato.

[36] porta ad avere un DCR direttamente proporzionale alla temperatura che puó essere fortemente diminuito al di sotto di 0^0C. Il problema principale é rappresentato dall' umiditá che porta alla formazione di condensa con il rischio di danneggiare la matrice di rivelatori a causa di corto circuito tra rivelatori. Per questo motivo la camera ermetica viene flussata con azoto in modo da evitare la formazione di condensa. In figura 1.14 é rappresentato un dettaglio della matrice di SPAD e circuiti AQC.

La linea tratteggiata bianca rappresenta la posizione della guarnizione che delimita il confine della camera ermetica. In figura 1.15 é invece rappresentata la vista dall' alto della camera completa di copertura e una sua prospettiva.

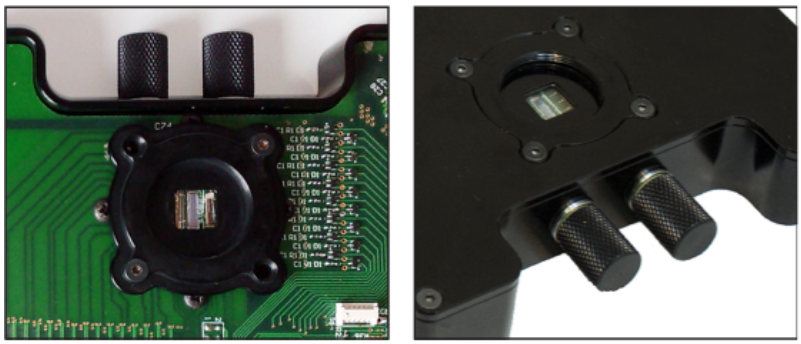

Figura 1.15: Immagine della camera ermetica vista dallálto e in prospettiva. In entrambe si notano le valvole di ingresso e uscita per flussare azoto all'interno della camera ermetica.

La scheda di elaborazione presenta due cammini paralleli come rappresentato in figura 1.16

per estrarre l'informazione di tempo e di conteggio di fotoni. Il percorso di timing (parte bassa della figura 1.16) é rappresentato da un collegamento verso un connettore parallelo mentre il percorso di counting prevede un collegamento verso FPGA per una elaborazione. Nel nostro progetto il ruolo di questa FPGA é semplicemente quello di ritrasmettere in uscita i segnali cosi come si presentano al suo ingresso in quanto l'elaborazione viene effettuata non sulla manta ma in una scheda collegata tramite cavo SCSI alla manta. Gli scopi di questi cammini sono differenti: il percorso di timing presenta delle specifiche di tempo stringenti ma non necessita di elevato numero di conteggi (al massimo qualche megacount ogni secondo) come quello di counting. Quest'ultimo invece ha richieste stringenti in termini di numero di conteggi ma non in termini di risoluzione temporale. Pertanto estrarre da un unico cammino in cui é presente l'FPGA per elaborazioni counting entrambe le informazioni porterebbe a caricare capacitivamente la linea di timing in quanto l' ingresso di FPGA é schematizzabile con una rete R C parallelo e quindi ad un peggioramento delle prestazioni soprattutto per quanto riguarda l'integritá del segnale di timing.

Scheda di alimentazione fornisce tutte le tensioni necessarie al corretto funzionamento di tutti gli elementi della manta. La manta é alimentata a 16V da cui sono derivate tutte le alimentazioni. La parte piú critica é

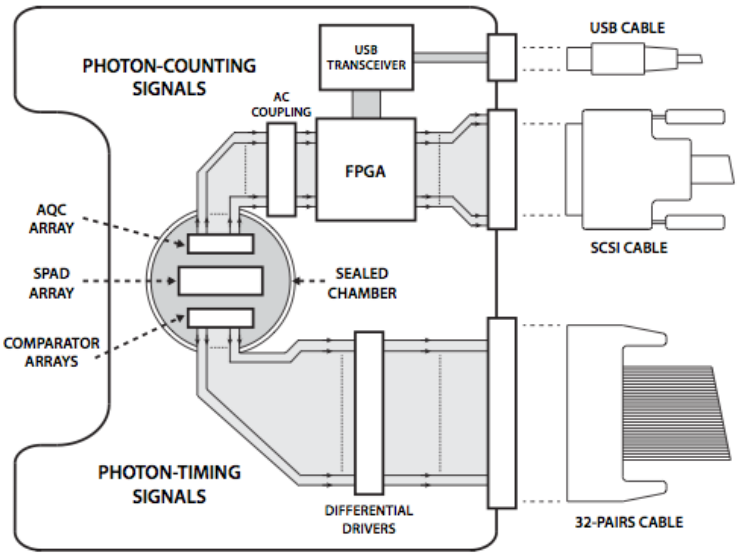

Figura 1.16: Schema a blocchi della scheda di elaborazione.

rappresentata dalla potenza dissipata della matrice di SPAD che richiede elevate tensioni di breakdown (ordine di -30, -40 V) nei quali scorre una corrente di valanga dell' ordine delle centinaia di milliampere. Inoltre la tensione di alimentazione degli SPAD deve essere stabile al fine di ridurre il jitter temporale.

Capitolo 2

Pogettazione hardware del sistema

Questo capitolo provvede a fornire una descrizione hardware del sistema realizzato il quale é costituito da due schede PCB: Board di front-end (scheda1) e Board principale (scheda 2). Verranno introdotti i componenti utilizzati insieme alle motivazioni che hanno portato alla loro scelta e verrá descritta la parte di alimentazione.

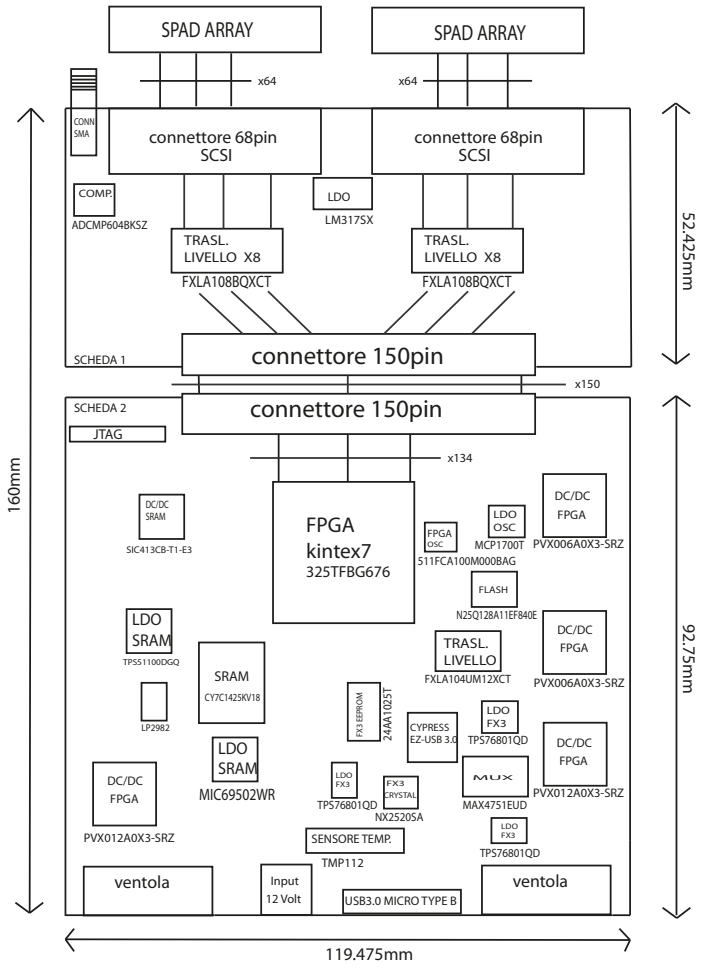

Figura 2.1: Schema completo delle due schede che mostra la disposizione dei componenti.

2.1 Scheda di front-end (1)

Scopo di questa scheda é quello di raccogliere i segnali provenienti dai rivelatori a singolo fotone SPAD. Tale scheda é composta da 4 layer ed ha una dimensone di 52.42 mm x 119.47 mm perché la dimensione totale, quando é collegata a quella principale, é tale da permetterle di entrare perfettamente nel package metallico.

La scheda di front-end é stata sviluppata su una PCB separata collegata alla scheda 2 tramite connettore a 150 pin (Samtec ERM-150) per permettere la massima flessibilitá del sistema, infatti nel caso si volesse acquisire dati da altri dispositivi che non utilizzano connetori SCSI, é sufficiente ridisegnare scheda di front-end con una piú adatta allo scopo, con un minimo sforzo in termini di riprogettazzione. Usando un connettore a 150 pin ma avendo 128 segnali da portare dai connettori SCSI abbiamo deciso di portare le alimentazioni dalla scheda 2 e di riservare ulteriori alimentazioni per future schede da collegare alla scheda 2.

- 64 segnali dallo spad A;
- 64 segnali dallo spad B;
- 2 segnali di START/STOP dalla manta A;
- 2 segnali di START/STOP dalla manta B;
- 2 segnali differenziali dal connettore SMA;
- 3 alimentazioni (12 V, 4.5V, 1.8V);
- la massa;

Come si puó osservare dall'elenco, vi é un segnale che giunge dal connettore SMA é serve per potersi sincronizzare con il sistema con il quale si sta effetuando la misura, ricevendo un segnale di clock da esso oppure inviandoglielo.

Dalla figura 2.1 si puó osservare che per collegarsi alle due mante, sono stati utilizzati due connettori SCSI da 68 pin ciascuno, ed ogni pin é stato

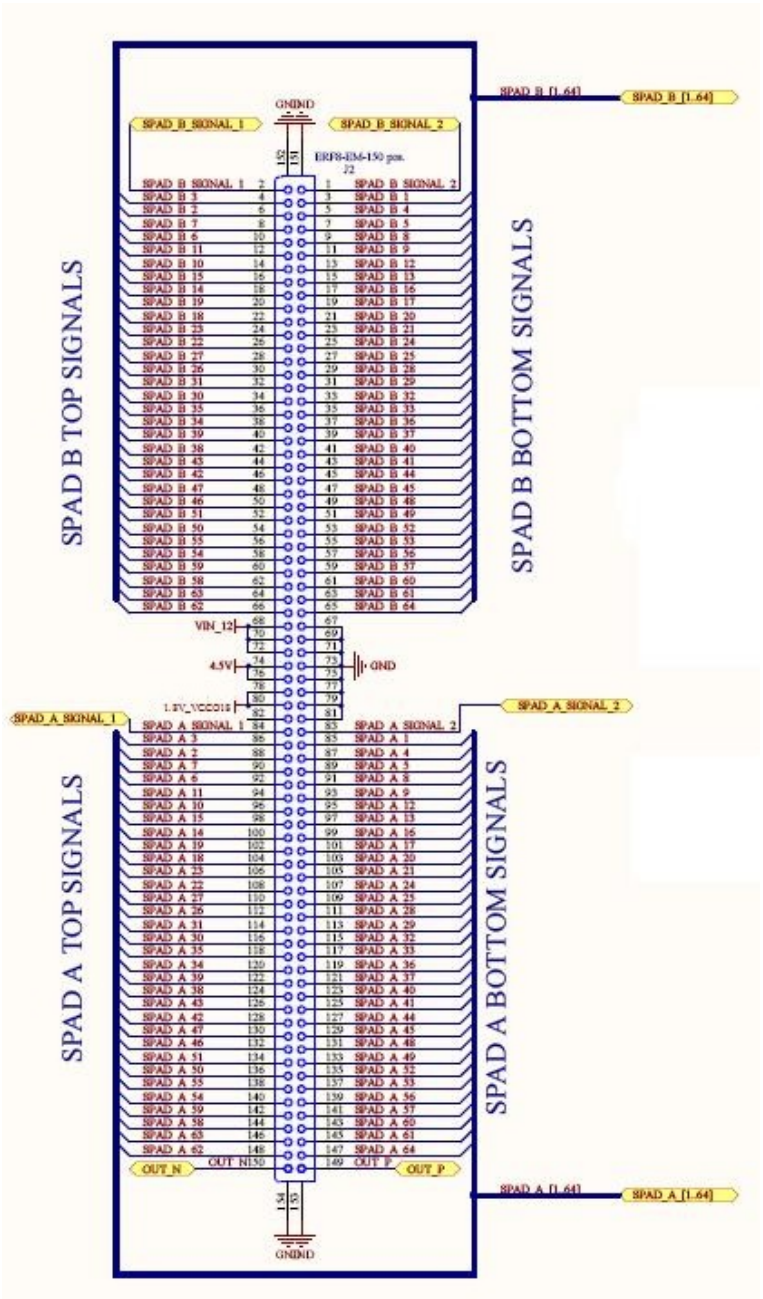

Figura 2.2: Schema di collegamento dei segnali che vanno dalle mante al connettore da 150 pin.

terminato con resistenze da 130Ω per adattarsi all' impedenza caratteristica del cavo SCSI. Tali connettori portano all'FPGA i segnali degli SPAD e di START/STOP alle mante. Successivamente questi segnali prima di essere inviati al connettore, passano attraverso dei traslatori di livello (FXLA108BQX della Fairchild Semiconductor [37]) i quali sono alimentati da un LDO della Texas Instruments (LM317SX/NOPB) che preleva la sua alimentazione di ingresso (12 V) dalla scheda 2. I segnali degli spad sono ad un potenziale di 3.3 V ma per poter essere letti dall'FPGA devono essere portati a 1.8 V dato che abbiamo scelto di alimentare i banchi I/O dell'FPGA a questa tensione. Abbiamo deciso di portare 1.8 V dalla scheda 2 e con essi di alimentare l'uscita dei traslatori di livello. La funzione di questi traslatori di livello é anche quella di rigenerare il fronte del segnale proveniente dallo SPAD in quanto tra esso e l'FPGA é presente una linea di trasmissione. Abbiamo scelto di routare i segnali sul top e sul bottom e di lasciare alimentazioni e massa sui layer interni. Questa scelta permette di ottenere una stessa capacitá vista dalle piste sul top e sul bottom in quanto risulta essere medesima la distanza tra il top layer e il layer interno (alimentazione) e tra il bottom layer e il layer interno (massa). Questo accorgimento serve a preservare l'integritá del segnale.

Per quanto riguarda il segnale che giunge dal connettore SMA, esso passa attraverso un comparatore, alimentato anch'esso a 3.3 V, che lo trasforma da single ended a differenziale e poi lo rende disponibile al connettore da 150 pin per essere gestito dall'FPGA.

2.1.1 L'ADCMP604

Tale comparatore, l' ADCMP604 della Analog Devices [38], riceve il segnale dal connettore SMA opportunamente terminato con una resistenza da 50Ω e lo trasforma da single ended a differenziale, cosí da poter fornire alla FPGA un clock esterno piú immune al rumore poiché ha una soglia per il rumore molto elevata. Per poter far ció, ha bisogno di una tensione di soglia, la THRESHOLD, che viene generata a partire dai 3.3V i quali sono poi partizionati tramite R135 e R137.

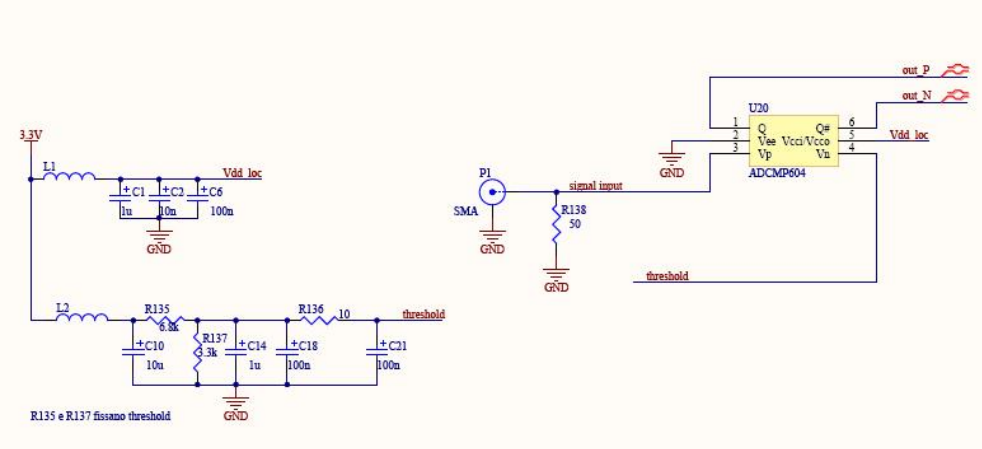

Figura 2.3: Schema di collegamento del comparatore e settaggio della relativa soglia.

2.2 Scheda principale (2)

E' stata realizzata su una PCB con 6 layer e la sua dimensone é di 92.75mm x 119.47mm. Questa scheda si occupa della elaborazione vera e propria dei segnali provenienti dalla scheda 1.

Ad essa é stato collegato un connettore da 150 pin, dei quali 132 portano i segnali che giungono dagli spad delle mante, 1 non é collegato e gli altri 15 le alimentazioni (12V, 4.5V 1.8V e GND) alla PCB di front-end. A questi segnali se ne aggiungono due differenziali che giungono dal connettore SMA (sulla scheda1) e servono per potersi sincronizzare con un segnale esterno come per esempio un clock. Al centro della scheda troviamo l'FPGA il quale é il cuore del sistema ed é il componente che riceve i segnali che giungono dagli SPAD e puó implementare il CROSS-CORRELATORE e il TIME-STAMPING entrambi a 128 canali. Esso é collegato ad una memoria FLASH (N25Q128A13ESE40E prodotta da Micron Technology), programmabile tramite connettore JTAG. L'FPGA puó avvalersi di questa memoria per immagazzinare la sua configurazione in modo che ad ogni sua accensione legga il bitstream che contiene il programma da svolgere. La scelta di questa memoria é stata fatta tenendo conto del numero di bit minimo che l'FPGA necessita per salvare i suoi dati di configurazione. Avendo a disposizione

piedini liberi nell'FPGA abbiamo sceto di inclusere nel nostro progetto anche una memoria SRAM (CY7C1425KV18-250BZC prodotta da Cypress), la quale al momento non é utilizzata, ma servirá per sviluppi futuri del progetto.

Il microcontrollore FX3 gestisce i trasferimenti tra scheda e PC interfacciandosi ponendosi come ponte tra l'FPGA e il PC in particolare abbiamo scelto il CYUSB3014-BZXI della Cypress poiché al momento dello sviluppo del nostro lavoro non erano presenti altri tipi di dispositivi sul mercato.

La necessitá di usare un dispositivo USB 3.0 anziché USB 2.0 deriva dal fatto che in primo luogo si voleva sviluppare un cross-correlatore a 128 canali che avrebbe richiesto uno scambio di dati notevole con il PC ad elevati throughput e quindi il protocollo USB2.0 non sarebbe stato sufficente a sostenere queste richieste. In secondo luogo, quando si é passati allo sviluppo del TIME-STAMPING, é sorta la necessitá di acquisire segnali con frequenza media di 100kHz su 128 canali e salvarli in 8 byte ciascuno. Il throughput risultante, 102MB/s, risulta del tutto insosostenibile per un protocollo USB 2.0.

2.2.1 FPGA

L'FPGA (acronimo di *Field-Programmable Gate Array*) é un dispositivo programmabile particolarmente apprezzato perché é possibile una sua programmazione e riprogrammazione in maniera flessibile. Aspetto questo, che lo rende adatto durante la fase di test per sviluppo di nuove applicazioni. Per il nostro progetto abbiamo deciso di utilizzare un FPGA prodotto da Xilinx in particolare il modello Kintex-7. La scelta si é basata sul confronto di alcuni report riguardanti le specifiche richieste per realizzare il cross-correlatore a 32 e a 64 canali. In particolare quelle che a noi interesavano di piú sono:

- block ram: si tratta della memoria dedicata volatile dell'FPGA
- slice: si tratta di celle logiche contenenti look-up table che svolgono le funzioni per cui é programmata l'FPGA.

Come si puó vedere dalle tabelle 2.1, 2.2 e 2.3, viene mostrato il confronto fra 3 FPGA: la XC6SLX150, la XC7K160T e la XC7K325T.

XC6SLX150	SLICE	BLOCK RAM 8kb	BLOCK RAM 16kb
VALORI TOTALI	23038	536	268
OCCUPAZIONE CROSS-CORRELATORE 32 CANALI	16389 71%	45 8%	1 1%

Tabella 2.1: Tabella nella quale sono indicati i valori totali dei fattori di interesse della SPARTAN 6 XC6SLX150 e i valori ottenuti dai report del cross-correlatore a 32 canali

XC7K160T	SLICE	BLOCK RAM 18kb	BLOCK RAM 36kb
VALORI TOTALI	25350	650	325
OCCUPAZIONE CROSS-CORRELATORE 32 CANALI	15937 62%	30 4%	8 2%
OCCUPAZIONE CROSS-CORRELATORE 64 CANALI	20698 81%	60 9%	16 4%

Tabella 2.2: Tabella nella quale sono indicati i valori totali dei fattori di interesse della KINTEX 7 XC7K160T e i valori ottenuti dai report del cross-correlatore a 32 canali e a 64 canali

XC7K325T	SLICE	BLOCK RAM 18kb	BLOCK RAM 36kb
VALORI TOTALI	50950	890	445
OCCUPAZIONE CROSS-CORRELATORE 32 CANALI	16965 33%	30 3%	8 1%
OCCUPAZIONE CROSS-CORRELATORE 64 CANALI	22471 44%	60 6%	16 3%

Tabella 2.3: Tabella nella quale sono indicati i valori totali dei fattori di interesse della KINTEX 7 XC7K325T e i valori ottenuti dai report del cross-correlatore a 32 canali e a 64 canali

Dai report e dalle tabelle precedenti, si puó osservare che per quanto riguarda l'occupazione di memoria, sia per quanto riguarda i 32 canali che i 64 canali, la richiesta non é stringente nei tre FPGA, poiché risulta essere molto bassa (massimo un 9%). I problemi sorgono quando si analizzano le SLICE necessarie.

Per la *SPARTAN 6* solo con il cross-correlatore a 32 canali si ottiene un utilizzo del 71% e quindi il passaggio a 64 canali risulterebbe critico e se si andrebbe oltre tale scelta risulterebbe totalmente inadeguata.

Passando alle *KINTEX 7* l'utilizzo percentuale di slice é inferiore alla SPARTAN 6 utilizzando un cross-correlatore a 32 canali, peró nel caso della XC7K160T passando al cross-correlatore a 64 canali si ottiene un' occupazione dell' 81% e questo valore é abbastanza elevato e non permette particolari sviluppi di codice VHDL in futuro. Se invece si analizzano i report della KINTEX 7 XC7K325T l'utilizzo percentuale di SLICE scende di molto infatti con il cross-correlatore a 32 canali é pari al 33% mentre con quello a 64 canali al 44%. Questo valore risulta essere moderato e inoltre permette notevoli sviluppi futuri in quanto é presente un sufficiente margine.

Quindi dalle caratteristiche appena elencate abbiamo deciso di scegliere la XC7K325T poiché rientrava nelle caratteristiche di nostro interesse.

Per ultimo abbiamo dovuto scegliere il package: FFG e FBG. Entrambe utilizzano la tecnologia *flip chip* e la principale differenza sta nei costi di produzione: la FBG é piú economica dal momento che é stata prodotta con meno passaggi e ció si traduce in un minor data rate e le performance termiche sono inferiori. Per il minor costo abbiamo deciso di scegliere il tipo FBG poiché il data rate é sufficiente per le nostre applicazioni e per quanto riguarda l'aspetto termico lo risolviamo utilizzando due ventole verticali che provvedono ad un riciclo d'aria all'interno del package in cui é alloggiato tutto il sistema.

Per i motivi sopra citati la scelta é ricaduta su una Kintex-7 XC7K325T-2FBG676C [39].

Analizzando meglio le caratteristiche della Kintex 7, si osserva che essa offre due tipologie di banchi I/O [40]:

- *high-performance (HP)*;
- *high-range (HR)*;

I primi sono stati progettati per andare incontro alle esigenze di performance di alta velocitá richiesta dalle memorie e da altre interfacce chip-to-chip con tensioni di 1,8V.

I banchi HR invece, sono progettati per supportare un maggior range di standards I/O con tensioni che possono arrivare a 3,3V.

A seconda di queste considerazioni sui banchi I/O abbiamo gestito in questo modo i segnali:

- banchi HP: abbiamo collegato i segnali da e per la SRAM poiché per essa abbiamo bisogno di essere piú performanti;
- banchi HR: abbiamo collegato i segnali da e per la FX3 e quelli che giungono dalle mante, poiché nel caso degli spad non ci sono in gioco frequenze elevate e i segnali hanno un range maggiore.

La comunicazione con la FX3 USB controller é implementata tramite una interfaccia slave FIFO sincorona a 100 MHz con bus dati a 16-bit. Grazie a questo bus dati parallelo, la FX3 permette all'FPGA di poter accedere direttamente al suo buffer interno rendendo molto semplice lo scambio dati. Per quanto riguarda la parte di alimentazione, abbiamo dovuto avere molta cura nella sua progettazione: in particolare sono state molto critiche le condizioni di operazione richieste ($1V \pm 3\%$ la piú critica). Per essere piú cautelativi abbiamo inserito dei filtri per i regolatori insieme ad una posizione strategica per evitare fenomeni di crosstalk tra i regolatori lineari e i buck che presentando un funzionamento switching potrebbero essere fonte di disturbi per i regolatori lineari. In particolare si é cercato di distanziare quanto piú possibile i regolatori lineari da quelli buck.

La Kintex-7 utilizzata ha in tutto 8 banchi I/O [40] (5 banchi HR e 3 banchi HP) ognuno dei quali ha 50 I/O.

A seconda dell'alimentazione del banco, ogni banco I/O puó supportare segnali di diversi standard come ad esempio LVDS, HCMOS ed inoltre puó implementare primitive come serdes, delayers o buffer.

Nel nostro caso abbiamo alimentato ogni banco con 1.8V poiché i segnali di comunicazione con la SRAM e la FX3 asseriscono ad uno standard di 1.8V e i segnali che giungono dalle mante sono stati opportunamente traslati in tensione a 1.8V dalla scheda di front end.

Per quanto riguarda le opzioni di configurazione dell'FPGA abbiamo scelto di optare per quella Master SPI configuration mode e con tale opzione va a caricarsi la configurazione direttamente dalla FLASH oppure nel caso fosse collegato il JTAG (il quale ha prioritá assoluta), lo si usa per configurare il dispositivo direttamente da PC [41].

Infine per generare il clock per l'FPGA abbiamo scelto di utilizzare il Si511 della Silicon Labs con uscita LVDS a 100 MHz.

2.2.2 Controllore USB 3.0

Come detto all'inizio, si é deciso di utilizzare il protocollo USB3.0 poiché quello USB2.0 non rientrava nelle specifiche di progetto. Dal momento che sviluppare il protocollo a livello hardware avrebbe impiegato un eccessivo sforzo e tempo, abbimo scelto di utilizzare un integrato, il EZ-USB FX3 prodotto dalla Cypress. Dopo vari test su demoboard che hanno dato esiti positivi, abbiamo deciso di includere il chip nel nostro progetto.

Il Cypress EZ-USB FX3 in figura 2.4 é stato progettato per supportare sia il protocollo USB 3.0 che quello USB 2.0 potendo sfruttare un bus dati a 32-bit (nel nostro caso usiamo un bus dati a 16-bit) ed un microprocessore ARM926EJ-S che opera a 200 MHz [42].

Per riuscire a sfruttare tutta la banda della USB 3.0, la FX3 é dotata di un'interfaccia, la GPIF II [43], la quale permette di gestire lo scambio dati fra PC e FPGA inoltre essa permette pure di interfacciarsi con altre periferiche in modo seriale sfruttando i protocolli SPI, I2C, I2S e UART. In particolare noi abbiamo utilizzato i primi due.

Per potersi configurare, la FX3, puó caricarsi il file immagine da diverse sorgenti e ció viene deciso a seconda di come sono stati impostati i pin di PMODE[2:0] [42] che sono mostrati in figura 2.5:

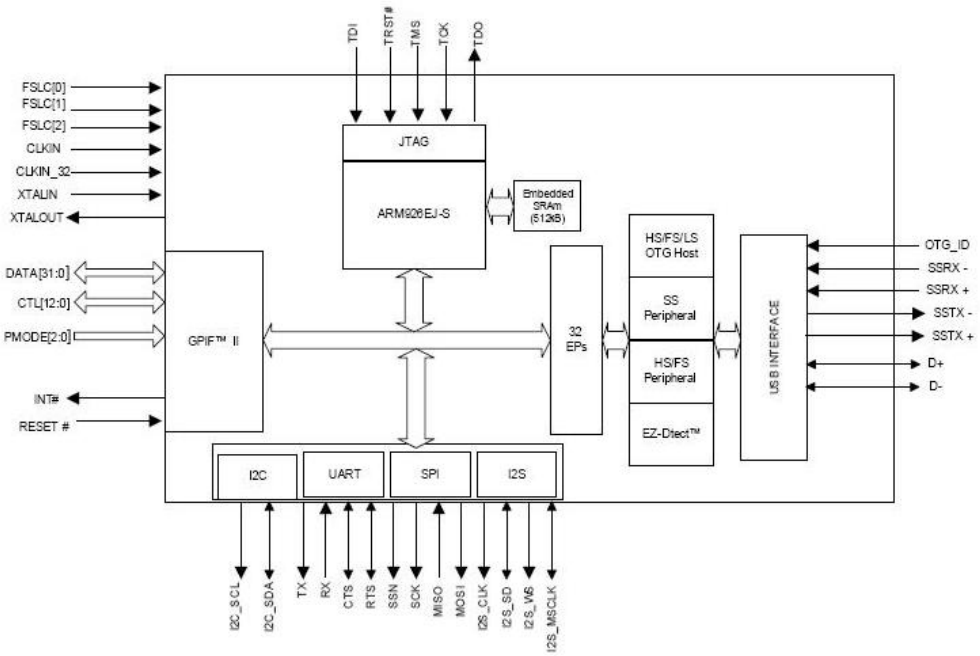

Figura 2.4: [42] schema a blocchi logico della Cypress EZ-USB FX3.

PMODE[2:0][2]	Boot From
F00	Sync ADMUX (16-bit)
F01	Async ADMUX (16-bit)
F11	USB boot
F0F	Async SRAM (16-bit)
F1F	I^2C, On Failure, USB Boot is Enabled
1FF	I^2C only
0F1	SPI, On Failure, USB Boot is Enabled

Figura 2.5: [44] possibili configurazioni di PMODE per le opzioni di BOOT.

Noi abbiamo impostato tali pin in modo da avere PMODE[F1F] e ció significa che il file immagine viene caricato da una memoria EEPROM da 1 Mbit collegata alla linea I2C, ma in caso di errore viene abilitato il boot.

Per generare il clock per il microprocessore é stato utilizzato un oscillatore al quarzo a 19.2 MHz, che poi tramite PLL, tale frequenza viene portata a 200MHz usati dalla logica interna della FX3. La frequenza a cui funziona la linea bus della FX3 (100MHz) viene fornita dalla FPGA [42].

Per quanto riguarda la parte di alimentazione della FX3[45] , quella del core é fissata a 1.2 V e quella di clock a 3.3 V mentre invece, viene lasciata abbastanza libertá di scelta per il resto dei banchi di I/O, in particolare si puó partire da 1.2 V fino ad arrivare a 3.3 V. Nel nostro caso abbiamo deciso di alimentare questi banchi a 1.8 V poiché tali pin di I/O devono comunicare con la FPGA, la quale parla a 1.8 V.

Per evitare problemi di overvoltage e di scariche elettrostatiche, la FX3 ha dentro di se una protezione ESD per il bus USB 3.0, per proteggerlo dalle scariche elettriche del corpo umano che si possono aggirare attorno ai $\pm 2.2kV$. Per essere piú cautelativi abbiamo aggiunto una protezione ulteriore, un componente ESD esterno , lo SP3010 che puó assorbire scariche anche di $\pm 15kV$ [45]. Passando alla parte di layout, sono state posizionate capacitá di decoupling il piú possibile vicino ai pin di ingresso, per evitare il propagarsi del rumore attraverso la parte di alimentazione.

Altre precauzioni sono state prese nel tracciare le 32 linee del bus dati, infatti il match peggiore fra le varie linee é di 1 µm quando quello raccomandato dal produttore [46] deve essere di almeno 12.7 mm e ogni linea é stata terminata con in serie un resistore da 22Ω. Questo é stato fatto per minimizzare problemi di timing, overshoot e effetti di ringing sulla linea di trasmissione.

Invece quando il segnale raggiunge la fine della linea di trasmissione, l'alta impedenza del ricevitore causa una riflessione e cosí il segnale che torna risulta raddoppiato rispetto all'inizio e quando la riflessione torna alla terminazione serie il potenziale lungo il resistore si annulla e quindi questo evita che ulteriore corrente entri nella linea. Dal punto di vista del ricevitore, questo effetto, fornisce una transizione logica perfetta senza overshoot e ringing.

Proseguendo con le tecniche di salvaguardia dell' integritá del segnale, é stato fatto un taglio nel layer subito sotto alle capacitá utilizzate per l'accoppiamento AC lungo la linea SuperSpeed differenziale di trasmissione dati

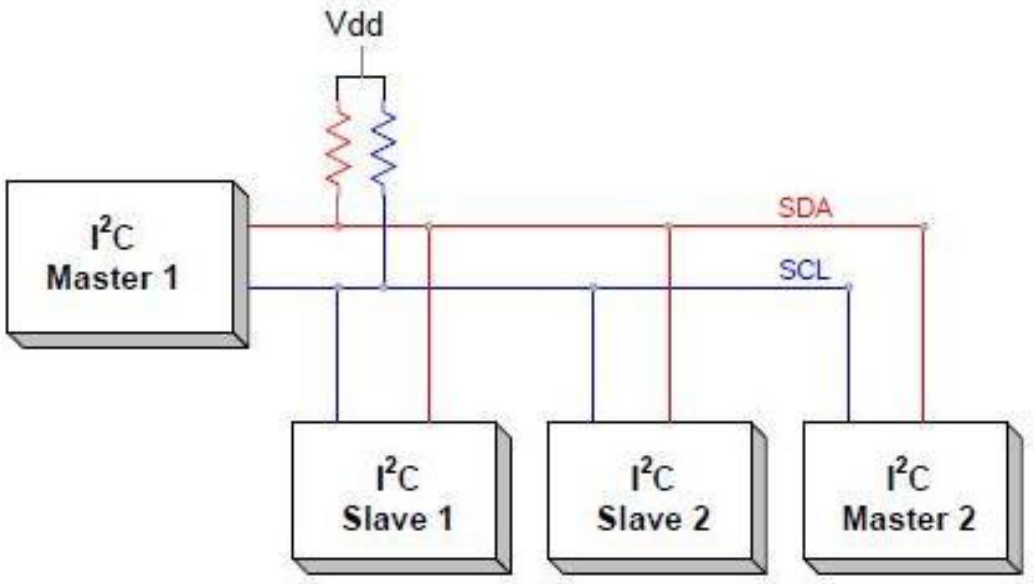

Figura 2.6: schema di collegamento di una classica rete I2C nella quale si possono notare due master del BUS e due slave.

come suggerito dall'user guide [46] relativa al layout della FX3. Ció é servito per evitare di avere capacitá parassite lungo la coppia di linee veloci che collegano il connettore alla FX3 [46].

Infine, come scelta del connettore, abbiamo deciso di utilizzare un connettore micro tipo B e le linee che dal connettore vanno alla FX3 sono state tracciate con molta cura, infatti vi é un match all'incirca di 5 µm , per evitare problemi nel trasferimento dati dal connettore all' interfaccia GPIF II.

2.2.3 Comunicazione seriale I2C

La comunicazione seriale I2C(in figura 2.6) é composto da almeno un *master* e uno *slave* (generalmente si ha un mastrer e piú slave, come nel nostro caso). Esso richiede due linee seriali di comunicazione:

- SDA (Serial DAta) per i dati;
- SCL (Serial CLock) per il clock (quindi é un bus sincrono).

Alle linee di segnale si aggiunge una linea di alimentazione a V_{dd} alla quale sono collegati i resistori di PULL-UP. Tale bus ha due tipi di nodi:

- nodo master: il dispositivo che emette il segnale di clock;
- nodo slave: il nodo che si sincronizza sul segnale di clock senza poterlo controllare;

all'inizio della trasmissione su bus I2C, il master inizia lo scambio di informazioni inviando lo START bit (S) seguito dall' indirizzo (7 bit da B1 a B7) dello slave con cui vuole comunicare. Poi segue un bit (B8) che indica se vuole trasferire informazioni allo slave o ricevere informazioni. Nel primo caso il bit B8 é tenuto basso dal master, invece nel caso voglia ricevere informazioni rilascerá la linea dati e andrá alto per la presenza del pull-up. Nel caso in cui lo slave indirizzato esistesse, prende il controllo della linea dati sul successivo impulso di alto del SCL e la forza bassa. Alla fine della trasmissione dati, viene inviato dal master lo STOP bit (S). L'FX3, tramite la sua interfaccia I2C puó solamente svolgere il ruolo di master del bus e fa ció ad una frequenza che puó andare dai 400 kHz fino ad 1 MHz con una tensione di $V_{dd} = 3.3V$ e comunica sia con il sensore di temperatura, il TMP112, che con la EEPROM 24AA1025T e proprio da quest'ultima, come detto precedentemente, allo start up, la FX3 si carica il firmware per configurarsi e poter essere operativa.

2.2.4 Sensore di temperatura

In fase di progetto abbiamo deciso di montare sulla nostra scheda un sensore di temperatura. A causa dell' elevata capacitá computazionale richiesta dall' FPGA e la presenza di numerosi regolatori di tensione i quali possono scaldarsi molto, per evitare di danneggiarli o in un caso peggiore di bruciarli é necessario monitorare la temperatura che permette di avere un'informazione in piú sullo stato della nostra scheda e puó essere utile per la gestione termica del sistema complessivo. Abbiamo quindi scelto l'integrato TMP112 di Texas Instruments. Questo sensore é alimentato a 3.3V e presenta una risoluzione di 12 bit. Questo dispositivo viene controllato attraverso un bus I2C dalla FX3.

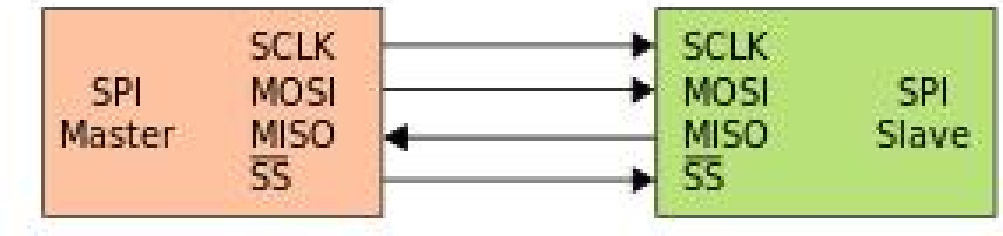

Figura 2.7: schema di collegamento di una classica rete SPI nella quale si possono notare un master e uno slave.

2.2.5 Memoria EEPROM

La EEPROM (acronimo di Electrically Erasable Programmable Read-Only Menory) é un tipo di memoria *non volatile*, usata nei dispositivi elettronici per memorizzare piccole quantitá di dati che devono essere mantenuti quando viene tolta l'alimentazione elettrica (per esempio la configurazione del dispositivo). Le operazioni di scrittura, cancellazione e riscrittura vengono fatte elettricamente.

Ciascuna cella di memoria capace di memorizzare un singolo bit é costituita da due transistori MOS, uno di memoria e uno di accesso. In fase di test tale memoria non é stata utilizzata. Versioni definitive del firmware verranno memorizzate in EEPROM.

2.2.6 Comunicazione seriale SPI

L'EZ-USB FX3, come detto precedentemente, tra le interfacce seriali, supporta la SPI (acronimo di Serial Peripheral Interface) in figura 2.7, che é un bus standard di comunicazione nel quale la trasmissione avviene tra un dispositivo detto *master* e uno o piú *slave*. Il master controlla il bus, emette il segnale di clock (quindi bus sincrono), decide quando iniziare e terminare la comunicazione. La comunicazione si basa su quattro segnali:

- MOSI (Master Output Slave Input): uscita per il master, input per lo slave;

- MISO (Master Input Slave Output): ingresso per il master, output per lo slave;
- SCLK (Serial CLocK): segnale di clock emesso dal master che puó arrivare a 33 MHz;
- CS# (Chip Select): serve ad abilitare lo slave e viene emesso dal master (si attiva a livello logico basso);

La trasmissione dei dati sul bus si basa sul funzionamento dei registri a scorrimento. Ogni dispositivo, sia master che slave é dotato di un registro a scorrimento interno, i cui bit vengono emessi e, contemporaneamente, immessi, rispettivamente, tramite l'uscita MOSI e l'ingresso MISO. Ad ogni impulso di clock, i dispositivi che stanno comunicando sulle linee del bus emettono un bit dal loro registro interno, rimpiazzandolo con un bit emesso dall' altro interlocutore.

La comunicazione viene intrapresa sempre su iniziativa del master che abilita lo slave tramite CS# e successivamente impone il clock sulla linea dedicata.

Con questa procedura ha inizio lo scambio dei bit tra i due shift register di master e slave. Alla fine di ogni parola trasmessa il contenuto del registro dello slave sará passsato al master e viceversa.

La FX3 utilizza questo bus, come anche la FPGA, per comunicare con la memoria FLASH.

2.2.7 Memoria FLASH

La FPGA normalmente viene configurata con il connettore JTAG, modalitá utilizzata soprattutto in fase di test per controllare se il codice VHDL sviluppato funziona correttamente. Questa programmazione presenta il vantaggio di cancellarsi a seguito dello spegnimento della FPGA. In una fase successiva, nel momento in cui si possiede la versione definitiva del codice, si andrá a memorizzarla sulla FLASH (in figura 2.8) cosí all'avvio, la FPGA andrá ad autoconfigurarsi direttamente da FLASH.

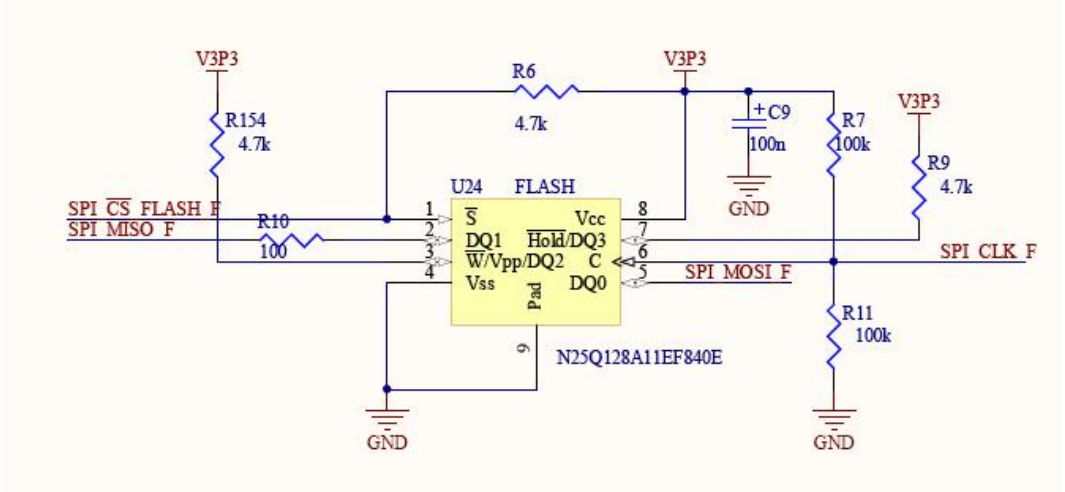

Figura 2.8: memoria flash utilizzata nella scheda principale nella quale si possono identificare i segnsali di MOSI, MISO, CS e CLK.

Per poter fare questo, bisogna impoostare opportunamente i pin di configurazione M0, M1 e M2 della FPGA [41]. A seconda di come vengono collegati questi pin, l'FPGA capisce come caricare il suo firmware, cioé se verrá programmata tramite JTAG, tramite flash o tramite un dispositivo parallelo. Collegando M0 all'alimentazione e M1 e M2 a massa, l'FPGA cercherá se é presente una configurazione in memoria FLASH.

Abbiamo scelto la FLASH N25Q128A della Micron [47] ed é alimentata a 3.3 V. Ha una capienza di 128 Mbit, la minima suggerita da Xilinx per contenere il suo bitstream di configurazione ma anche sufficiente per i nostri scopi.

2.2.8 Multiplexer

Il MULTIPLEXER (MAX4751 prodotto dalla Maxim Integrated [48]) é costituito da una serie di 4 interrutttori che mettono in comunicazione o scollegano i contatti posti ai suoi estremi, come si vede in figura 2.9. Questo dispositivo ci serve perché noi vorremo mettere in comunicazione la FLASH sia con la FX3, che con la FPGA.

La memoria FLASH é sempre collegata alla FPGA per permetterle all'accensione di scaricarsi la sua configurazione. Nel caso in cui si volesse, in

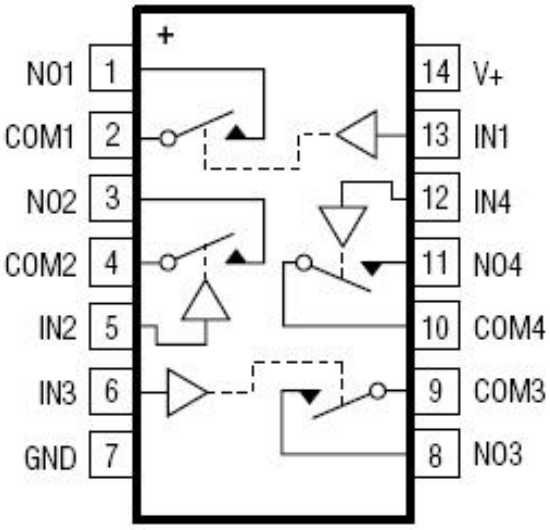

Figura 2.9: schema interno del multiplexer [48].

un secondo momento, modificare il codice VHDL, diventa necessario riprogrammare la FLASH. Per questo scopo, si deve far uso della FX3, ma per poterlo fare si deve inibire la FPGA dal bus SPI per evitare conflitti con la FX3. A tale scopo abbiamo deciso di utilizzare un multiplexer e abbiamo collegato ad una sua estemitá i segnali del bus SPI che giungono dalla FX3 e all' altra estremitá i segnali del bus SPI che giungono dalla FPGA come indicato in figura 2.10. La memoria FLASH é sempre collegata alla FPGA poiché allo start-up o al reset del sistema, la FPGA si possa riconfigurare in automatico.

Il processo di riconfigurazione della FLASH e successivamente della FPGA segue questi passi:

1 : la FX3 tramite il segnale PROGRAM_B disattivsa momentaneamente la FPGA;

2 : la FX3 tramite ON/OFF SPI comanda il multiplexer di modo che, metta in comunicazione la FLASH e la FX3;

3 : la FX3 riconfigura la memoria FLASH;

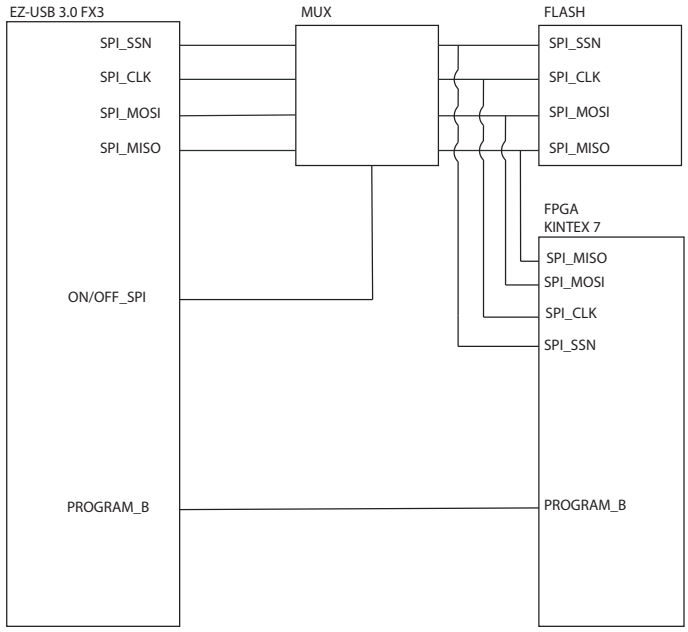

Figura 2.10: schema di collegamento tra FLASH, FX3 e FPGA gestito dalla FX3 tramite un multiplexer.

4 : la FX3 tramite ON/OFF SPI si scollega dalla FLASH;

5 : la FX3 rilascia PROGRAM_B e la FPGA si riconfigura con il nuovo codice scritto sulla FLASH;

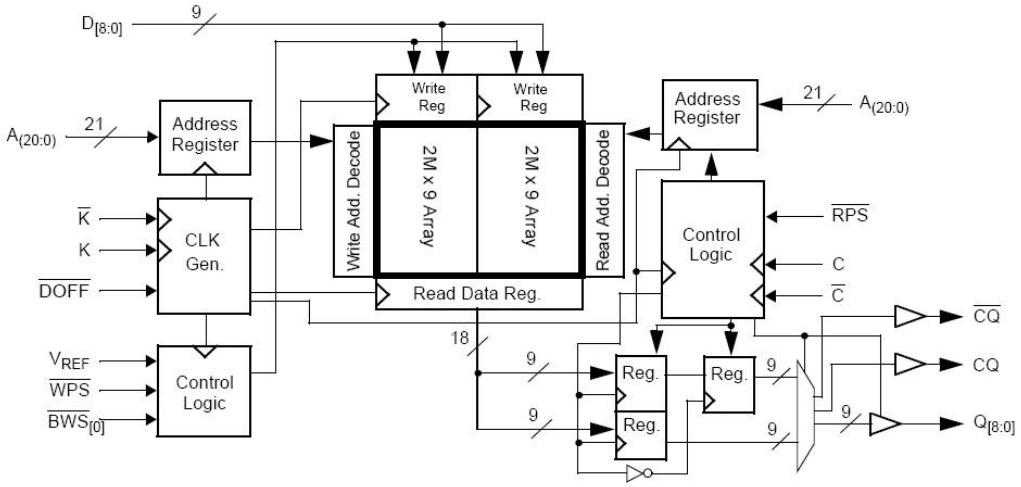

Figura 2.11: schema logico a blocchi della SRAM [49].

2.2.9 SRAM

La memoria SRAM utilizzata, é prodotta dalla cypress [49] ha una capienza di 36 Mbit organizzati in 9 banchi da 4 M parole da 9 bit ciascuna e puó operare fino a 333 MHz. Appartiene alla serie dual port QDRII (Quad Data Rate di seconda generazione) e puó trasferire quattro dati per ciclo di clock: due in lettura e due in scrittura.

Come si puó vedere dallo schema dei segnali, si puó osservare che essa possiede due ingressi di clock differenziali, rispettivamente K e C, e fornisce a sua volta in uscita il clock CQ.

Il clock K viene utilizzato dalla memoria per campionare gli ingressi (comandi, dati ed indirizzi) e fornisce i dati in uscita se si opera in Single Clock Mode, altrimenti quest' ultimi vengono allineati all'altro ingresso di clock C. L'altro clock, CQ, é centrato sui dati in uscita in modo che il controllore li possa campionare correttamente.

I comandi di lettura e scrittura vengono registrati utilizzando gli ingressi #RPS (Read Port Select) e #WPS (Write Port Select) rispettivamente. Il bus indirizzi é composto da 23 linee anche se le parole sono 2^{24} e ció é possibile perché ad ogni accesso viene letta e/o scritta una parola in ciascun banco. Gli ingressi e le uscite sono in logica HSTL (High Speed Transceiver Logic)

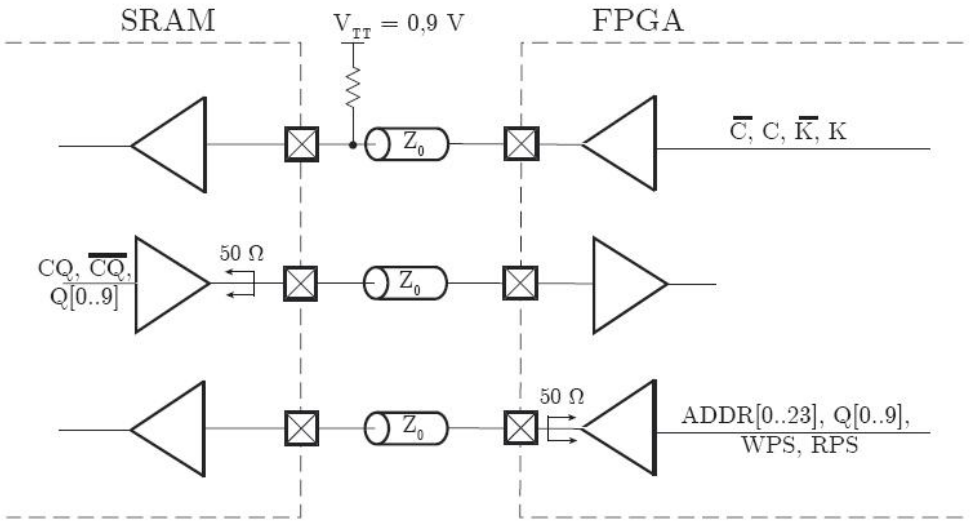

Figura 2.12: schema delle terminzaioni verso VTT [49].

il cui valore alto é definito dall'alimentazione VDDQ dei buffer d'uscita che puó essere 1.5 V o 1.8 V (nel nostro caso é 1.8 V).

L' ingresso VREF anch'esso a metá dinamica, é utilizzato per impostare la soglia dei buffer d'ingresso.

I segnali HSTL, normalmente vengono terminati lato carico con una resistenza verso VTT, come si vede in figura 2.12 il cui valore é VDDQ/2 per mantenere una simmetria in fase di commutazione e per tale motivo i clock C e K sono stati terminati verso VTT, vicino alla SRAM .

Escludendo i clock, il numero di linee di segnale é considerevole, quindi si é deciso di utilizzare le terminazioni sorgente interne ai due componenti figura 2.13. Si é deciso di adattare alla linea le impedenze dei buffer d'uscita sia della SRAM che dell' FPGA. Per l'FPGA consiste nella configurazione in fase di programmazione, mentre per quanto riguarda la SRAM viene utilizzato come riferimento la resistenza esterna collegata al pin ZQ.

Successivamente abbiamo dovuto tracciate le linee in modo da avere un'impedenza controllata per ognuna di esse rispettando in linea di massima;

- $Zo = 50\Omega$
- $Zd = 100\Omega$

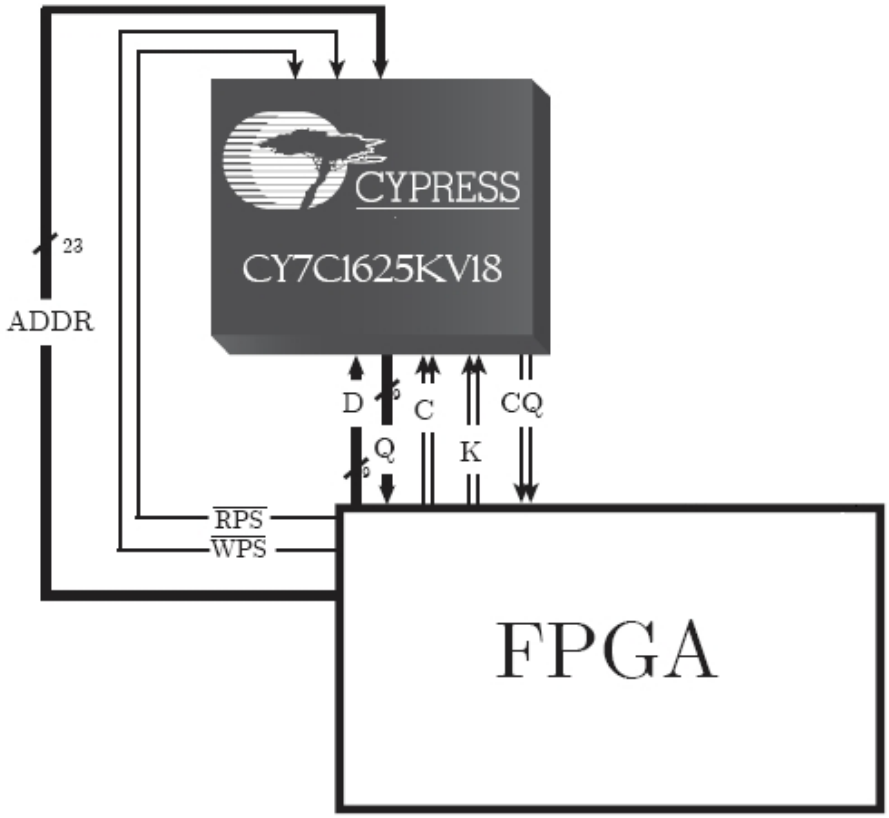

Figura 2.13: schema di collegamento tra FPGA e SRAM con i vari segnali.

Inoltre abbiamo curato che il piano subito sotto a quello dove passano le linee di segnale, sia sul TOP che sul BOTTOM, sia un piano di GND o di alimentazione in modo che non ci siano problemi per quanto riguarda le correnti di ritorno, poiché a tali frequenze le linee di segnale si devono considerare a parametri distribuiti. In tale modo, ad ogni commutazione il segnale che si propaga nella linea ha un'ampiezza dimezzata, ma si riflette completamente una volta arrivato al carico non terminato, che di fatto vede l'intera dinamica del segnale originario.La riflessione poi, termina il suo percorso una volta tornata alla sorgente.

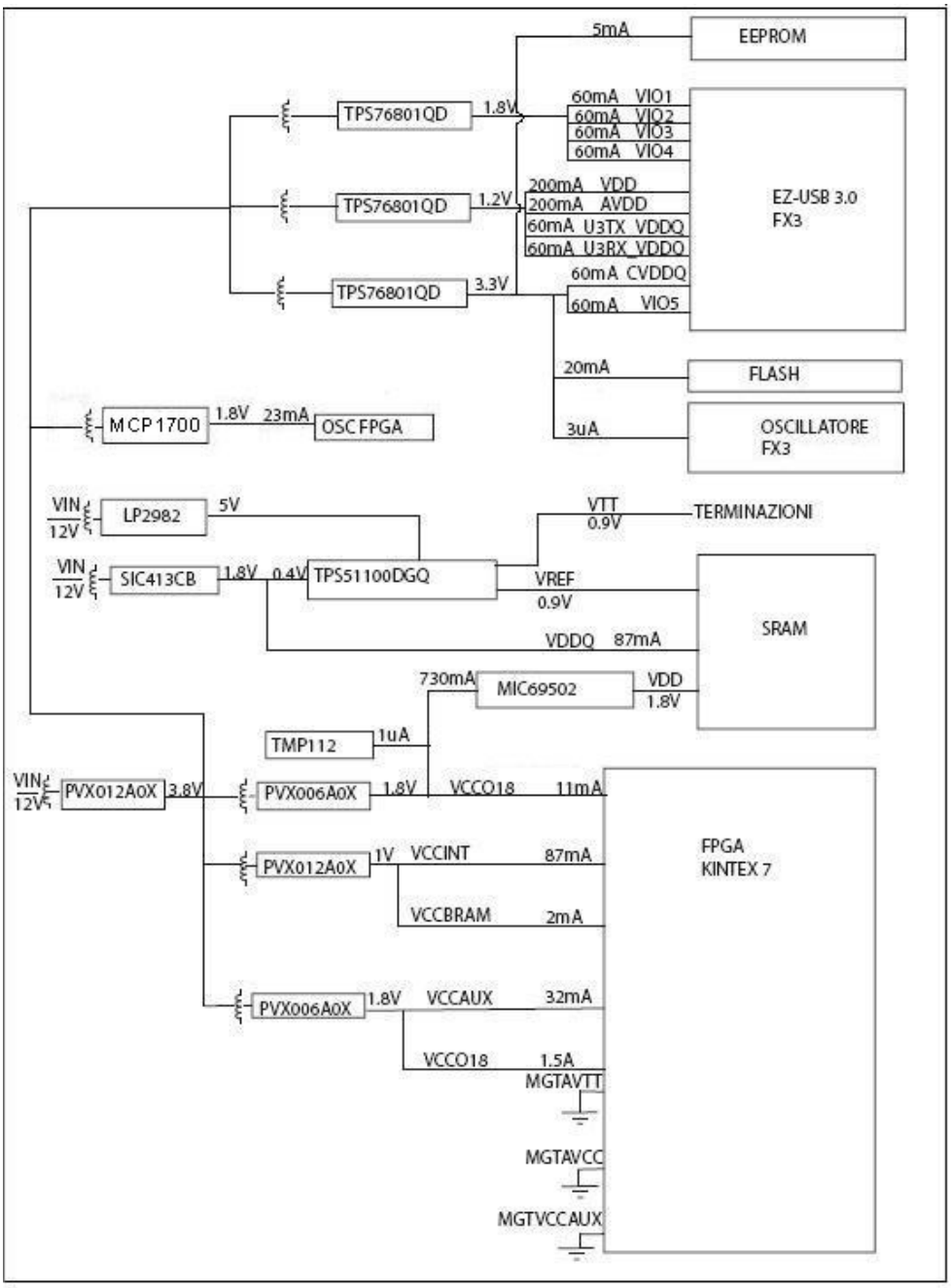

Figura 2.14: schema della rete di alimentazione della scheda principale.

2.3 Rete di alimentazione

Uno schema completo della rete di alimentazioene della scheda princiale é mostrato in figura 2.14.

L'intero sistema é pensato per essere alimentato da un adattatore esterno AC/DC, il quale fornisce una tensione di 12 V e una corrente massima di 5 A. La rete di alimentazione dell'FPGA é stata dimensionata usando le stime ottenute da un programma fornito dalla Xilinx (Xilinx Power Estimator) che permette di avere una idea delle correnti massime richieste dalle varie alimentazioni dell'FPGA fissato il carico computazionale previsto. Poiché la massima tensione richiesta dal sistema é di 5V, i 12 V sono convertiti ad una tensione piú bassa in modo che sia possibile aumentare l'efficenza dell'intero sistema.

2.3.1 FPGA

Scegliere i componentri che alimentano la FPGA é stato critico poiché richiede l'uso di numerose tensioni differenti ed é il componente che all' interno del sistema presenta il maggiore consumo di potenza. Le tensioni da fornire sono [50]:

- V_{CCINT}: alimentazione della logica interna;
- $V_{VCCBRAM}$: alimentazione dei blocchi di RAM interni;
- V_{CCO} : alimetazione ricihiesta da ogni banco per gestire gli input e gli output;
- V_{CCAUX}: alimentazione di vari blocchi di logica di interconnessione e dei buffer d'ingresso per alcuni standard di tensione sia single-ended che differenziali;

La tensione V_{CCO} é stata scelta pari a 1.8 V poiché comunica con tutti i dispositivi utilizzando standard a questa tensione e ove non sia cosí abbiamo utilizzato dei traslatori di tensione per adattarla al valore corretto. Le altre

	Tensione nominale (V)	Massima variazione tollerata	Corrente massima stimata
V_{CCINT}	1.0	3%	87 mA
V_{CCBRAM}	1.0	3%	2 mA
V_{CCO18}	1.8	5%	1.51 A
V_{CCAUX}	1.8	5%	32 mA

Tabella 2.4: Tabella nella quale vengono indicati i valori nominali delle tensioni di alimentazione della FPGA [50], le massime variazioni tollerate e la corrente massima assorbita stimata dal tool *Xilinx Power Estimator*.

tensioni della FPGA hanno valori nominali fissati dal datasheet e sono tutte illustrate in tabella 2.4.

La fase di scelta dei regolatori é stata fatta in seguito allo studio dell'evaluation board KC705 basata sulla Kintex 7. In quel caso, la rete di alimentazione é composta da regolatori switching da cui abbiamo tratto spunto per il nostro progetto. Abbiamo deciso di adottare moduli di regolatori switching comprensivi di induttori e programmabili analogicamente attraverso resistori esterni. Per fornire V_{CCINT} e V_{CCBRAM} abbiamo utilizzato un unico regolatore switching PVX012A0X di General Electric [51], un unico modulo PVX006A0X [52] per V_{CCAUX} e V_{CCO18} e un altro modulo PVX006A0X per la V_{CCO18}.

Questi regolatori soddisfano a pieno le specifiche su ogni rail di alimentazione in quanto garantiscono una regolazione di carico di 10 mV a fronte delle massime variazioni di corrente tollerabili da ciascuno (da 0 A a 12 A per il PVX012A0X e da 0 A a 6 A per il PVX0060X).

La tensione d' ingresso di questi regolatori é pari a 3.8 V ed é generata da un ulteriore PVX012A0X che a sua volta ha in ingresso i 12 V di alimentazione. Tale riduzione della tensione d'ingreso permette di migliorare l'efficienza (P_{OUT}/P_{IN}) dei 3 regolatori in cascata facendone quindi diminuire la potenza dissipata.

2.3.2 Controllore EZ-USB FX3

Pr quanto concerne l'alimentazione della FX3, abbiamo deciso di usare gli stessi LDO consigliati sul datasheet dell'EZ-USB FX3 ovvero i TPS76801QD della Texas Instruments, i quali generano le tensioni di: 1.2 V, 1.8 V e 3.3 V.

- 1.8 V: alimentano 4 domini indipendenti di I/O per poter comunicare con gli altri dispositivi sulla scheda, poiché tutti comunicano con standard a questa tensione;
- 1.2 V: alimentano il core logico dell'interfaccia USB 3.0, la parte analogica, il PLL e il cristallo oscillatore;
- 3.3 V: alimentano il clock interno e uno dei cinque dominii I/O che serve per comunicare con il bus I2C.

Come si osserva dalla tabella 2.5 le tensioni massime assorbite dalla FX3, a parte quelle assorbite dal core V_{DD} e A_{VDD}, non sono particolarmente elevate, per cui questo componente non ha una parte di alimentazione particolarmente critica e in piú avendo portato la tensione di ingresso di questi LDO a 3.8 V permette di abbassare notevolmente la $V_{in} - V_{out}$ ai loro capi e di conseguenza la potenza dissipata risulta notevolmente contenuta.

2.3.3 Memoria EEPROM

La memoria EEPROM necessita di una alimentazione di 3.3V e la corrente massima richiesta é di 5 mA e poiché viene utilizzata per la configurazione della FX3 e comunica con essa tramite bus I2C che é ad una tensione di 3.3 V, abbiamo deciso di collegarla al TPS76801QD che fornisce i 3.3 V alla FX3.

2.3.4 Sensore di temperarura

Il sensore ha bisogno di un' alimentazione di 3.3 V e la corrente richiesta é di 1 µA.

Tensione nominale (V)	Alimentazione	Corrente massima assorbita (mA)
1.2	V_{DD}	200
	A_{VDD}	200
	$U3TXV_{DDQ}$	60
	$U3RXV_{DDQ}$	60
1.8	V_{IO1}	60
	V_{IO2}	60
	V_{IO3}	60
	V_{IO4}	60
3.3	C_{VDDQ}	60
	V_{IO5}	60

Tabella 2.5: Tabella nella quale vengono indicati i valori nominali delle tensioni di alimentazione della FX3 [45] e inoltre vengono indicate le massime correnti assorbite come indicato sul datasheet .

Poiché la richiesta di corrente é esigua abbiamo deciso di alimentare questo componente con il LDO TPS76801QD che fornisce i 3.3 V alla FX3, dal momento che comunica con essa tramite bus I2C.

2.3.5 Memoria FLASH

La memoria FLASH necessitá di un' alimentazione di 3.3 V, nonostante comunichi con la FPGA e la FX3 su bus SPI a 1.8 V, poiché in commercio non vi erano memorie disponibili con le caratteristiche di alimentazione di nostro interesse. Dal momento che comunica su bus SPI, abbiamo utilizzato traslatori di livello che portano i segnali della FLASH da 3.3V a 1.8V in

modo che possano essere compatibili. Dal momento che la richiesta massima di corrente é inferiore ai 20 mA [49] abbiamo deciso di collegarla all' LDO TPS76801QD che fornisce i 3.3 V alla FX3.

2.3.6 Oscillatore FX3

L'oscillatore della FX3 viene alimentato direttamente dall' EZ-USB FX3 mediante il dominio dedicato ed assorbe una corrente molto esigua di 3 µA.

2.3.7 Oscillatore FPGA

L'oscillatore della FPGA richiede un' alimentazione di 1.8V e richiede una corrente di 23mA.

Dal momento che la richiesta di corrente é molto ridotta, abbiamo deciso di alimentarlo con il MCP17000 e nonostante richieda poca corrente, abbiamo deciso di non collegarlo al PVX0060X che alimenta i banchi della FPGA per evitare che quest' ultimo introduca disturbi sul clock della FPGA.

2.3.8 SRAM

La memoria SRAM richiede quattro diverse tensioni di alimentazione [49]:

- V_{DD}: é l' alimentazione del CORE ed é di 1.8V. Da datasheet si legge che non deve subire variazioni maggiori di 100 mV e la massima corrente richiesta é di 950 mA;
- V_{DDQ}: é l'alimentazione dei buffer di uscita e si trova a 1.8V, poiché essa comunica con la FPGA la quale ha tutti i banchi I/O alla medesima tensione;
- V_{REF}: é l'alimentazione di riferimento che si trova a 0.9V e puó subire variazioni non superiori ai 72 mV dal momento che serve alla memoria per distinguere i livelli logici in ingresso;
- V_{TT}: é l'alimentazione per le terminazioni ed é pari a 0.9V.

Lo standard HSTL utilizzato dalla SRAM per comunicare dagli altri dispositivi, necessita di una resistenza di terminazione di 50 Ω fra la linea e un livello di tensione pari a metá dell'alimentazione dei buffer di uscita. La FPGA puó terminare internamente a $V_{CCO18}/2$, dove tale tensione é quella del banco di I/O della FPGA, quindi se scegliamo di alimentare i buffer a 1.8V, questa terminazione interna risulta corretta.

Per capire come alimentarla al meglio, abbiamo analizzato un' evaluation board di ALTERA: *100G Develelopment Kit* [53]. Dall' analisi abbiamo visto che la V_{DDQ} per i buffer viene generata dal buck LTM4601, per generare la V_{DD} il lineare MIC69502 e invece per la V_{TT} e la V_{REF} si utilizza il TPS51100.

Per generare la V_{DDQ} noi abbiamo scelto di utilizzare il SIC413CB della Vishay Siliconix [54], poiché ha una buona regolazione di carico pari allo 0.6% ovvero 11 mV e quindi rientra nelle specifiche indicate sopra, inoltre prende in ingresso 12 V e li abbassa a 1.8 V.

L'alimentazione del core, abbiamo deciso di generarla come visto dall' evaluation board, utilizzando il MIC69502 di Micrel, che ci permette di fornire 5 A con regolazione di carico massima pari allo 0.2%.

Anche per la V_{TT} e la V_{REF} abbiamo seguito quanto utilizzato nella evaluation board, utilizzando il TPS51100DGQ di Texas Instrumentation che puó erogare una corrente fino a 3 A. Per funzionare necessita di due tensioni: 5 V e 1.8 V. Per fornire la prima abbiamo utilizzato l' LP2982, un lineare di piccole dimensioni poiché la richiesta di corrente da parte del TPS51100DGQ é inferiore ai 2 mA. Per quanto riguarda gli 1.8 V, sono stati forniti direttamente dal SIC413CB.

2.3.9 Scheda di front-end

Per quanto riguarda la scheda di front end, essa riceve le alimentazioni dalla scheda principale, in particolare tramite il connettore da 150 pin, riceve la GND, i 12V e gli 1.8V e inoltre abbiamo riservato anche alcuni piedini liberi in cui portiamo 4.5V. Quest'ultima tensione non é usata nel nostro progetto ma nel futuro si potranno collegare nuove schede di front-end che necessitano di tale alimentazione.

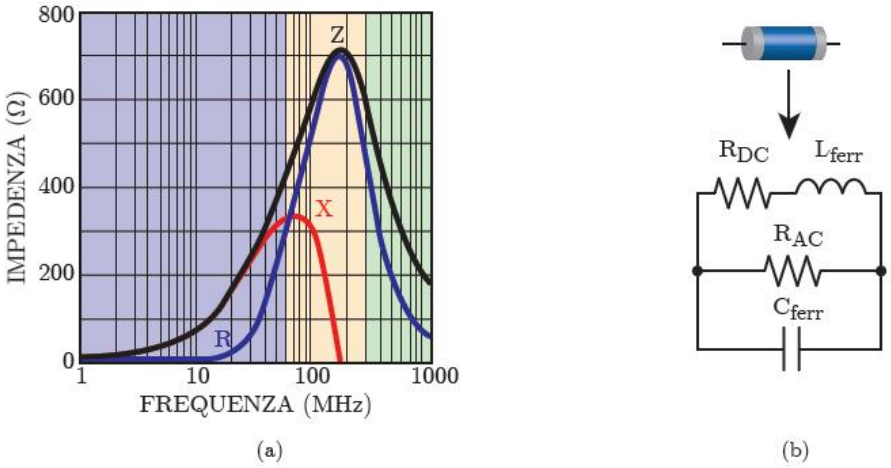

Figura 2.15: (a) andamento in frequenza dell'impedenza di una ferrite. Si possono notare le tre diverse bande in cui cambia il comportamento dell'impedenza. (b) modello circuitale single branch della ferrite

2.3.10 Progettazione di filtri con ferriti

Teoria sulle ferriti

Nelle reti di alimentazione complesse, per evitare il propagarsi di rumore e di crossralk tra linee di alimentazione diverse, é buona norma utilizzare capacitá di by-pass e posizionarle molto vicine ai pin di alimetazione. In tale modo si riducono le componenti AC spurie e i ripple di tensione causati dai grandi transitori di corrente. Per aumentare l'effetto filtrante e creare dei filtri passa basso da mettere in serie alle alimentazioni, vengono usate le ferriti.

Le ferriti sono un tipo di induttori con nucleo magnetico costituito da un composto di materiali ceramici come MnZn o NiZn. L'andamento in frequenza dell'impedenza di una ferrite é mostrato nel grafico ZRX fornito dai costruttori e lo si puó osservare in figura 2.15.

Dal grafico si possono distinguere tre diverse regioni di funzionamento: nella regione a bassa frequenza presenta basse perdite per isteresi e per correnti parassite, invece a media frequenza (nell'esempio da 60 MHz a 300 MHz), si osserva un comportamento resistivo, quindi i disturbi filtrati in questa zona sono dissipati sotto forma di calore. Infine ad alta frequenza ha

un comportamento capacitivo a causa degli accoppiamenti tra le spire.

Progetazione di un filtro con ferriti

Come esempio di progettazione prendiamo la ferrite applicata all'alimetazione del MCP1700, che é colui che genera la tensione per l'oscillatore dell'FPGA.

Prima di cominciare con il progetto del filtro dobbiamo considerare i seguenti requisiti:

- massima variazione dell'alimentazione tollerabile: non essendo indicata sul datasheet, la scegliamo in modo arbitrario, $\Delta V = 0.1V$;
- corrente massima richiesta: tale alimentatore, alimenta solo l'oscillatore, il quale richiede $I_{max} = 23mA$;
- banda di attenuazone: l'oscillatore genera una sinusoide a 100 MHz, quindi il limite inferiore é scelto come $f1 = 100MHz/10 = 10MHz$, mentre per quanto riguarda il limite superiore, poiché non ho specifiche, viene scelto in modo arbitrario e pari a $f2 = 700MHz$;
- attenuazione in banda: esso viene scelto pari a -40 dB.

Ora passiamo alla progettazione della rete di filtraggio passo passo:

- Molto importante é ricavare l'impedenza massima in banda d'attenuazione vista dal carico (Z22). Per ottenerla abbiamo utilizzato il software PDNTool di Altera [55] che ci permette di studiare il comportamento in frequenza delle capacitá di decoupling pari a 1 uF (3 da 0.33 uF). Con la simulazione della C di decoupling, abbiamo potuto osservare che il valore massimo in banda d'attenuazione é di 8Ω a 500 MHz quindi l'aggiunta della ferrite non deve aumentare l'impedenza vista dal carico, per cui la Z22 in banda d'attenuazione deve essere inferiore a 8Ω.
- La Rdc é la massima resistenza in continua che la ferrite puó avere e la si ricava considerando la massima corrente richiesta all'alimentazione e il ripple tollerabile: $RDC = \Delta Vtoll/I_{max} = 0.1V/23mA = 4.35\Omega$.

- Per ottenere i 40 dB di attenuazione in banda, é necerssario che l'impedenza della ferrite sia almeno 100 volte maggiore di quella della rete di decoupling. Quindi la minima impedenza della ferrite a 500 MHZ deve essere : $Zfer@500MHz = 100 \cdot 8\Omega = 800\Omega$;

- Vogliamo attenuare di 40 dB in banda d'attenuazione e il passa basso ha un doppio polo, quindi la frequenza di taglio risulta essere:

  $Fco = 10^{-40/40} \cdot f1 = 10^{-1} \cdot 10MHz = 1MHz$

- A bassa frequenza il termine dominante nell'impedenza Z22 é dato dall'induttanza della ferrite. Se imponiamo che Z22 sia pari a 8Ω otteniamo: $Z22 = jwL_{ferrmin} = 8\Omega$ quindi $L_{ferrmin} = 8\Omega/2 \cdot \pi \cdot 1MHz = 1.27uH$

- In base alle specifiche sopra calcolate, é stata scelta la ferrite MH1608-600Y della Burns, che ha le seguenti caratteristiche :

  - $Zfer@700MHz4$;
  - $L_{fer} = Z_{fer}@40MHz/40MHz = 40/2 \cdot \pi x40MHz = 0.159uH$;
  - $RDC = 40m\Omega$;
  - $I_{max} = 3A$

- I condensatori di by-pass (Cbp) hanno una capacitá complessiva pari a 1 uF e quindi la frequenza di taglio del filtro é:

  $Fco = 1/2 \cdot \pi \cdot \sqrt{L_{fer} \cdot Cbp)} \simeq 400kHz$

  che é accettabile, quindi non é necessario aggiungere altri condensatori per cambiare la Fco.

2.4 Considerazioni meccaniche

I segnali che dalle mante giungono al connettore da 150 pin, nella scheda di front end, che poi nella scheda principale proseguono verso l'FPGA, sono segnali che hanno una frequenza tale per cui le linee di trasmissione sulle quali viaggiano non si possano considerare linee a parametri concentrati ma bensí a parametri distribuiti. Onde evitare problemi di cross-talk, sono state tracciate linee di trasmissione ad impedenza controllata, in particolare con $Zo = 50\Omega$ e $Zd = 100\Omega$ ed in piú, come si vede dalla figura 2.16 sono state tracciate, ove possibile, sul TOP e sul BOTTOM per fare in modo che il piano sotto di esse possa essere di GND o di alimentazione, ma soprattutto omogeneo, ossia senza tagli o interruzioni di modo che le linee possano sperimentare tutte la stessa capacitá rispetto al loro piano di riferimento.

Questo discorso vale anche per le linee di trasmissione tra FPGA e SRAM e tra FPGA e FX3, come giá citato in precedenza, poiché utilizziamo segnali a frequenze tali da dover considerare le linee a parametri distribuiti e quindi per evitare disturbi e ritardi nei segnali trasmessi, anche sotto di esse abbiamo cercatro il piú possibile di tracciare piani omogenei. A causa di un limitato numero di layer non é sempre stato possibile fare questo.

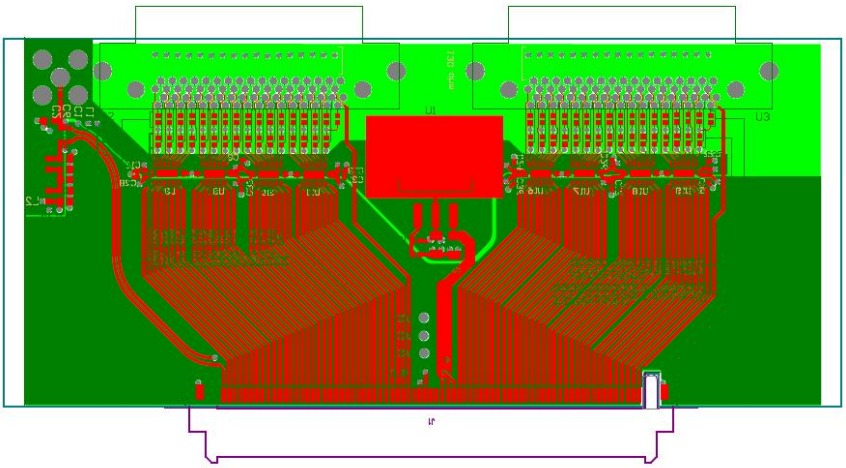

Figura 2.16: immagine di layout del TOP della scheda di front-end che mostra il fatto che le linee sono tracciate tenendo una distanza costanze l'una dall'altra e sotto di esse é presente un piano di alimentazione contiguo per evitare il piú possibile problemi di crosstalk .

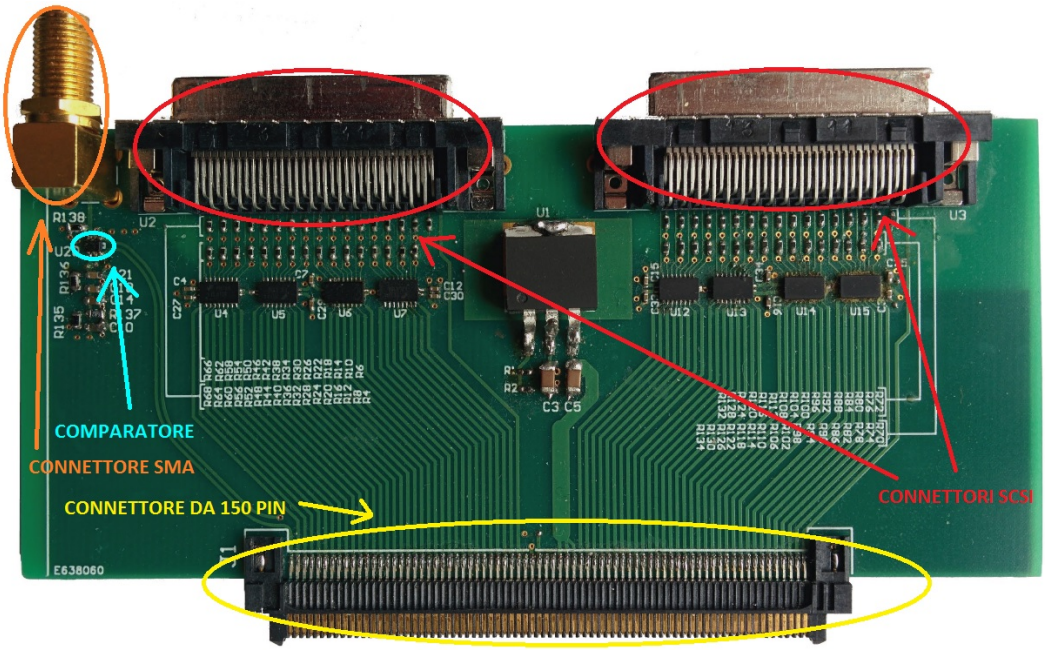

Figura 2.17: immagine del TOP della scheda di front-end che mostra le linee di segnale che vanno dal connettore a 150 pin alla FPGA. Come si puó vedere, ove possibile, sono state tracciate parallele ed equidistanti le une dalle altre, per ottenere Z_o e Z_d controllate ed evitare il piú possibile problemi di crosstalk fra le linee.

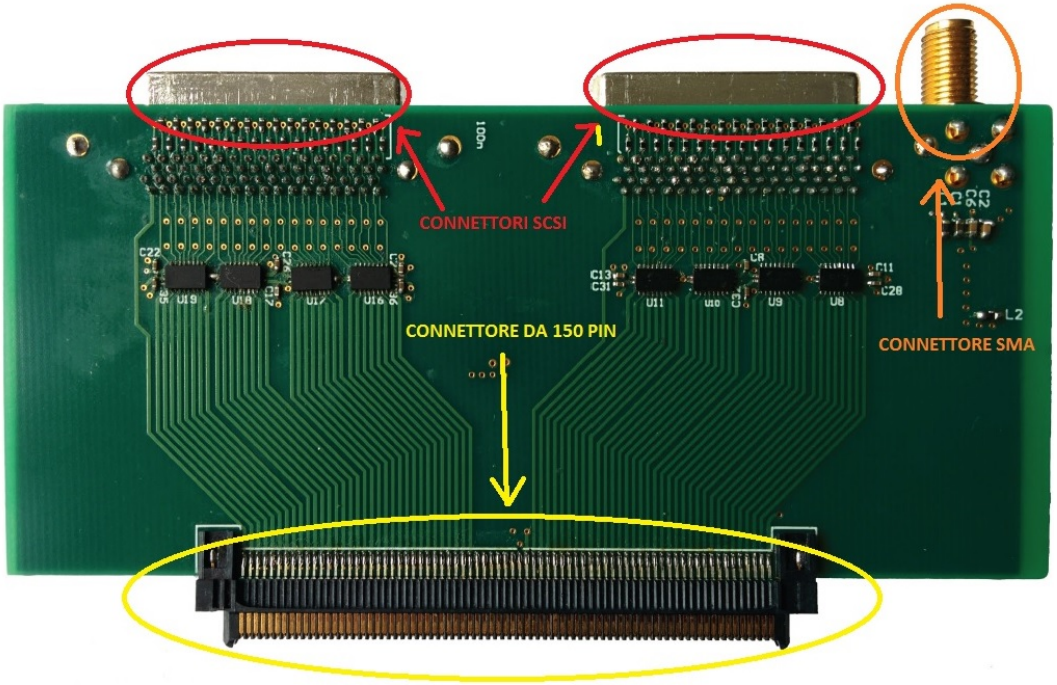

Figura 2.18: immagine del BOTTOM della scheda di front-end che mostra le linee di segnale che vanno dal connettore a 150 pin alla FPGA. Come si puó vedere, ove possibile, sono state tracciate parallele ed equidistanti le une dalle altre, per ottenere Z_o e Z_d controllate ed evitare il piú possibile problemi di crosstalk fra le linee.

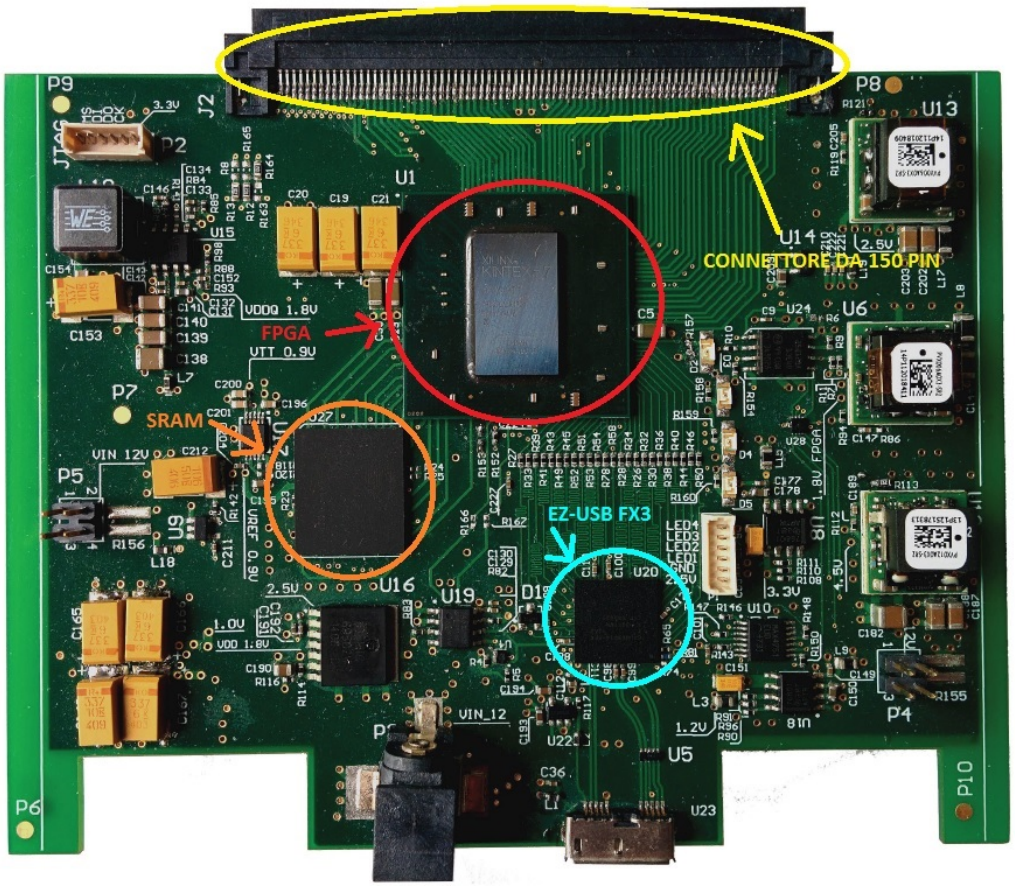

Figura 2.19: immagine del TOP della scheda principale che mostra le linee di segnale che vanno dal connettore a 150 pin alla FPGA. Come si puó vedere, ove possibile, sono state tracciate parallele ed equidistanti le une dalle altre, per ottenere Z_o e Z_d controllate ed evitare il piú possibile problemi di crosstalk fra le linee.

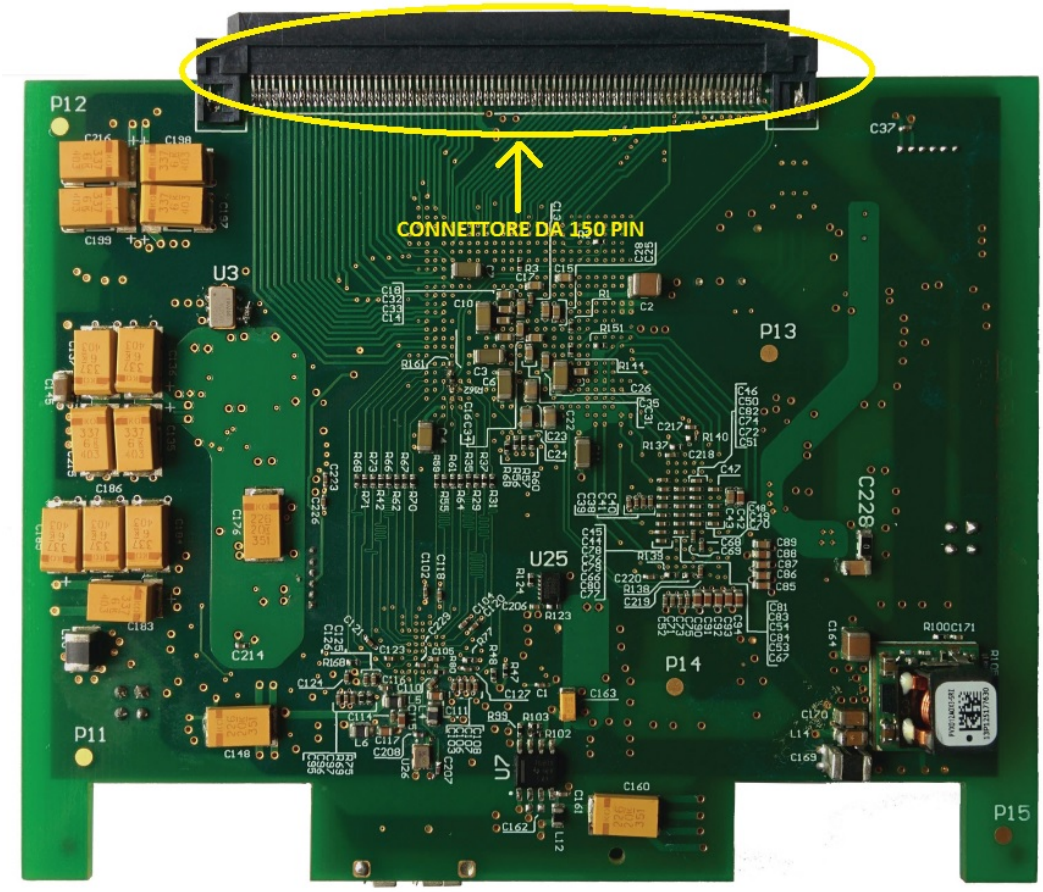

Figura 2.20: immagine del BOTTOM della scheda principale che mostra le linee di segnale che vanno dal connettore a 150 pin alla FPGA. Come si puó vedere, ove possibile, sono state tracciate parallele ed equidistanti le une dalle altre, per ottenere Z_o e Z_d controllate ed evitare il piú possibile problemi di crosstalk fra le linee.

Capitolo 3

Firmaware e software

In questo capitolo tratteremo in dettaglio l'implementazione della macchina a stati che realizza il protocollo USB3.0 e gestisce lo scaricamento dati verso PC. Descriveremo i test fatti e i software sviluppati per verificare la correttezza del VHDL implementato ed infine tratteremo i firmware e software creati per applicazioni time stamping.

3.1 Firmware VHDL

Per comunicare con la EZ-USB FX3, l'FPGA si interfaccia con la GPIF II la quale utilizza la modalitá SLAVE FIFO [56] e permette di accedere direttamente alla memoria interna della FX3 per svolgere operazioni di lettura o scrittura di dati.

Di seguto sono elencati i segnali dell'interfaccia e in figura 3.1 ne viene mostrata la direzione (verso FX3 verso FPGA o entrambe):

- SLCS#: questo é un segnale attivo basso e serve ad abilitare il chip;
- PKTEND#: questo é un segnale attivo basso, e serve per selezionare la modalitá di invio short packet o zero lenght packet;
- SLWR#: questo é un segnale attivo basso e la FPGA lo deve attivare per avviare una sessione di scrittura dati verso FX3;

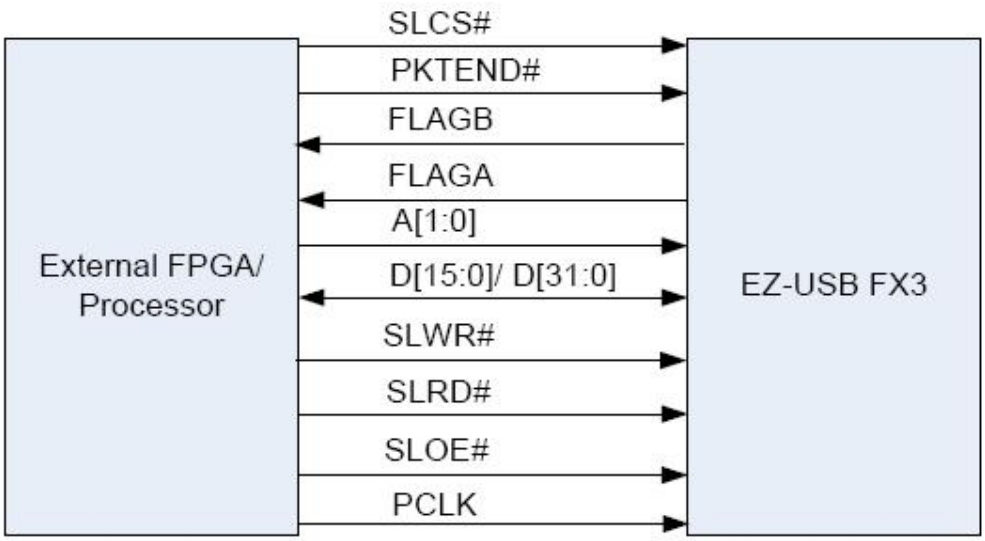

Figura 3.1: Schema di collegamento fra FX3 e FPGA con i segnali e la corrispondente direzione per la modalitá di comunicazione SLAVE FIFO [56].

- SLRD#: questo é un segnale attivo basso e la FPGA lo deve attivare per avviare una sessione di lettura dati da FX3;
- SLOE#: questo é un segnale attivo basso e la FPGA lo deve attivare per rendere disponibile il bus dati alla FX3 durante la fase di lettura dati da quest'ultima;
- PCLK: a questo pin va collegato il clock dell' FPGA per permettere la sincronizzazione;
- FLAGA/FLAGB: flag che indicano la disponibilitá di un socket della DMA;
- A[1:0]: 2 bit che servono ad indirizzare il socket corretto;
- D[31:0]: data bus da 32 bit;

A causa di un errore nella fase di layout, sono disponibili solo 16 delle 32 linee dato. Il progetto é stato comunque portato a termine usando un bus ridotto in quanto questo limita solo il massimo rate raggiungibile. In

uno sviluppo futuro questo errore verrá corretto. Prima di passare a trattare l'implementazione del processo di trasferimento dati tra FX3 e FPGA spiegheremo meglio il concetto di DMA .

3.1.1 Il DMA

Il DMA [42] é un meccanismo che permette ad altri sottoinsiemi, quali ad esempio le periferiche, di accedere direttamente alla memoria interna per scambiare dati, in lettura e/o scrittura, senza coinvolgere l'unitá di controllo per ogni byte trasferito tramite l'usuale meccanismo dell'interrupt e la successiva richiesta dell'operazione desiderata, ma generando un singolo interrupt per blocco trasferito.

Peer la parte USB3.0 é fondamentale per poter raggiungere elevate velocitá di trasferimento dati. Esso consente di mettere direttamente in comunicazione l'interfaccia GPIF II con la sezione USB, senza passare dalla CPU (ARM9). In questo modo il flusso di dati é continuo e non viene interrotto continuamente dal processore, ma é proprio il DMA ad occuparsi di questo.

Come si puó vedere dalle figure 3.2, 3.3, 3.4, 3.5 essa é costituita principalmente da 4 elementi :

- Socket;
- Descrittori;
- Thread;
- Buffer;

I socket sono punti di connessione tra una periferica, come la GPIF II, e la RAM della FX3 ed includono un set di registri i quali contengono i descrittori del DMA. Un socket che scrive dati nella RAM dell'FX3 viene chiamato producer socket, mentre uno che legge i dati é chiamato consumer socket.

I descrittori del DMA contengono informazioni circa gli indirizzi e le dimensioni dei buffer del DMA, che sono una parte della RAM totale della FX3: la dimensione massima di un buffer é di 64 kbytes.

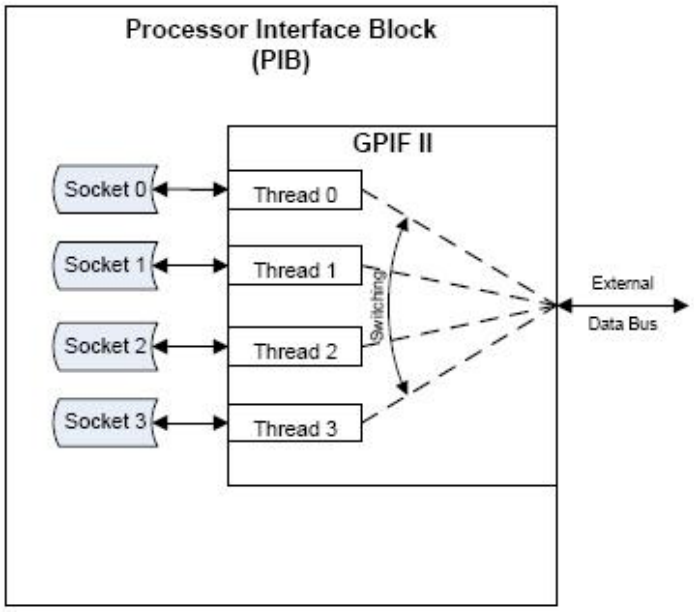

Figura 3.2: mappatura di default dei SOCKET/THREAD dell'interfaccia GPIF II [42].

I thread della GPIF II, sono dei percorsi dati che connettono i pin esterni ad un socket. La EZ-USB FX3 dispone di 4 threads fisici per poter scambiare dati tramite la GPIF II e ognuno puó essere indirizzato utilizzando i segnali A[1:0].

Le condizioni di buffer vuoto, parzialmente vuoto, pieno o parzialmente pieno sono segnalate con i flagA e flagB, che possono essere associati a thread correnti o dedicati: nel primo caso i thread fanno sempre riferimento allo stesso socket, mentre nel secondo caso devono essere indirizzati tramite i due bit di address A[1:0] per accedere al socket voluto.

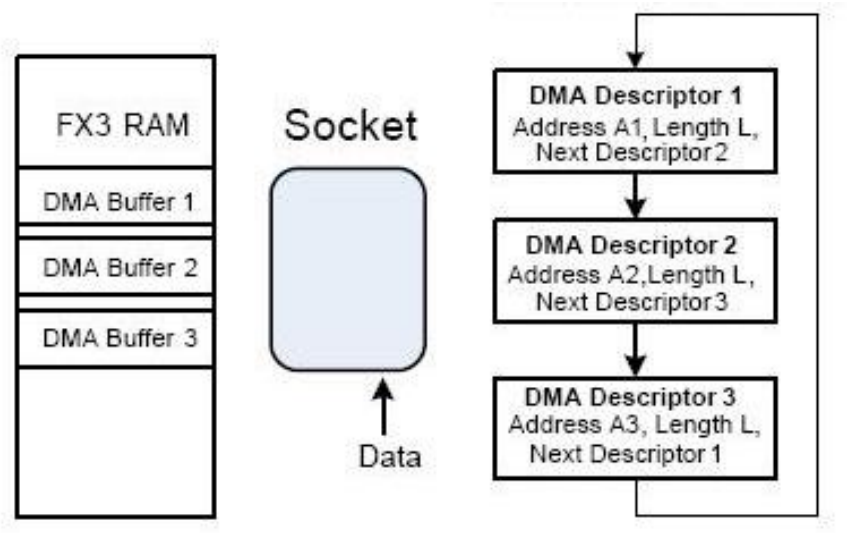

Figura 3.3: sulla destra é presente una lista di descrittori del DMA, al centro il socket d'interesse per trasmettere i dati e a sinistra la RAM della FX3 nella quale la DMA scrive direttamente i dati, utilizzando appositi buffer [42].

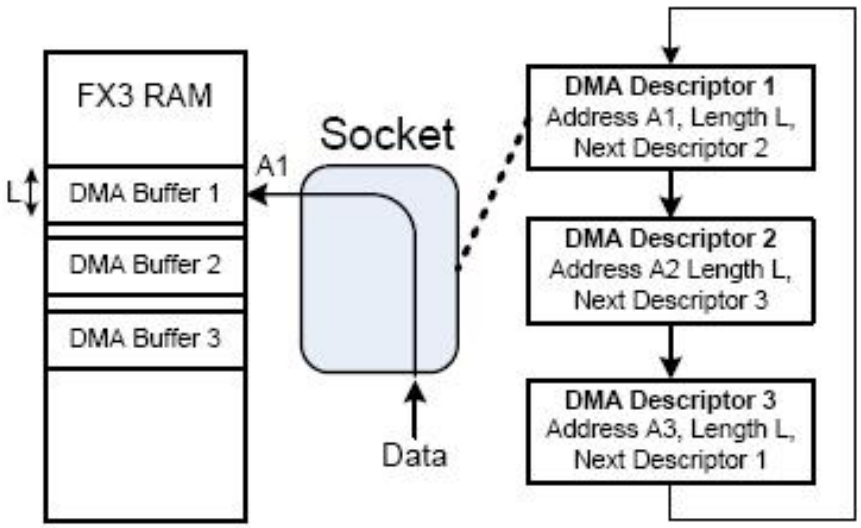

Figura 3.4: il socket ha caricato il descrittore 1 della DMA, il quale indica che si deve trasferire L bytes partendo dall'indirizzo A1 e nel frattempo va ad interrogare il descriuttore 2 della DMA [42].

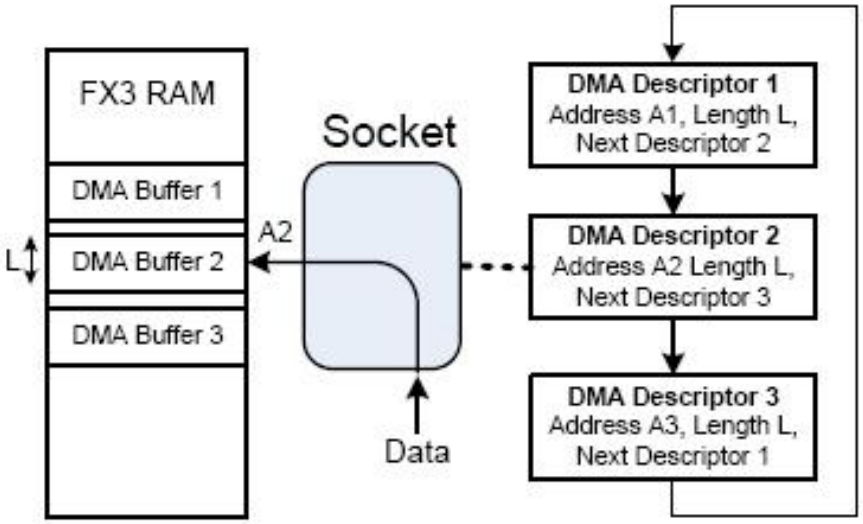

Figura 3.5: in questo passaggio vengono svolte le operazioni descritte dal descrittore 2 del DMA nello stesso modo della figura 3.4 [42].

3.1.2 Trasferimento dati: la macchina a stati

Come si vede in figura 3.1, l'FPGA ha bisogno di ricevere i segnali dalla FX3 di FLAG A (full flag dedicato al thread 0 che indica quando il buffer é pieno o vuoto) e di FLAG B per sapere quando quest'ultima ha i buffer disponibili per la ricezione o il trasferimento di dati. Per poter gestire il trasferimento dati fra i due dispositivi, si é fatto uso di una macchina a stati che prevede due macchine a stati secondarie:

- *STREAM OUT*: in questo stato l'FPGA attende la ricezione di dati (comandi) dalla FX3 che li ha ricevuti a sua volta da PC tramite connettore USB3.0.
- *STREAM IN*: in questo stato l'FPGA invia dati alla FX3 (per esempio risultati di misure, segnali di controllo o altro) per fare in modo che poi vengano instradati al PC ed in seguito analizzati con opportuni software che tratteremo piú avanti.

Prima di proseguire con la trattazione passo-passo della macchina a stati é opportuno precisare che il FLAG B non é stato utilizzato, nonostante sia stato

introdotto precedentemente. Il perché di questa scelta é dovuto al fatto che l'asserimento di questo flag soffre di una latenza. Di conseguenza si é preferito utilizzare un contatore interno per monitorare lo stato di occupazione dei buffer. Da qui in avanti utilizzeremo sempre questo metodo, anche se in alcune parti capiterá di vedere qualche rimando al FLAG B, ma questo é stato fatto solo per mantenere gli stessi nomi consigliati dalla casa costruttrice della FX3.

Un'ultima precisazione riguarda il fatto che nella singola macchina a stati dello STREAM OUT compaiono le diciture di FLAG C e FLAG D. A livello hardware questi sono collegati agli stessi pin di FX3 e di FPGA che competono rispettivamente a FLAG A e FLAG B della macchina a stati dello STREAM IN. L'unica cosa che li distingue é l'indirizzamento che viene fatto a livello di VHDL tramite i due bit A[1:0] che nel caso dello STREAM IN sono $A[1:0] = 0$ mentre nel caso dello STREAM OUT sono $A[1:0] = 3$ e quindi fanno riferimento a due socket diversi, quello 0 e quello 3.

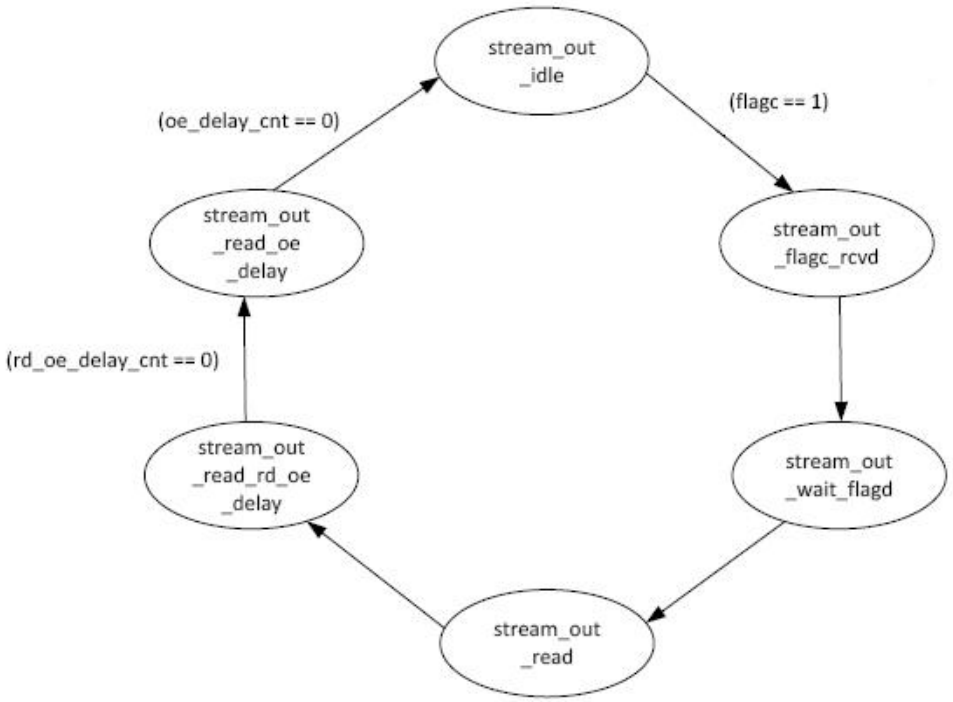

Figura 3.6: viene mostrata la macchina a stati dello STREAM OUT stato per stato [56].

3.1.3 Stream out

Per poter capire meglio come funziona la macchina a stati dello stream out, facciamo riferimento allo schema a blocchi in figura 3.6 fornito dalla Cypress [56].

Lo STREAM OUT parte dallo stato iniziale STREAM_OUT_IDLE con i segnali di comando asseriti in questo modo:

- $PKTEND\# = 1$;
- $SLOE\# = 1$;
- $SLRD\# = 1$;
- $SLCS\# = 0$;
- $SLWR\# = 1$;

- $A[0:1] = 3$;

Quando $FLAGC = 1$ lo stream out passa allo stato STREAM_OUT _FLAGC_RCVD e passato un ciclo di clock, passa allo stato STREAM_OUT _WAIT_FLAGD. Una volta qui, passato un altro ciclo di clock, la macchina a stati passa allo stato STREAM_OUT_READ, e l'FPGA imposterá i segnali di comando nel modo seguente:

- $PKTEND\# = 1$;
- $SLOE\# = 0$;
- $SLRD\# = 0$;
- $SLCS\# = 0$;
- $SLWR\# = 1$;
- $A[0:1] = 3$;

In questo modo l'FPGA riceve i dati dall'FX3 tramite il bus a 16 bit e decodifica il comando ricevuto da PC. Terminata la trasmissione dati, il FLAG C passa dallo stato HIGH al LOW per indicare che il buffer é *completamente riempito*, lo STREAM OUT passa allo stato STREAM _OUT_READ_RD _OE_DELAY e qui vi rimane per un colpo di clock.

Successivamente passa a STREAM_OUT_READ_OE_DELAY e in questo stato i segnali di controllo vengono modifcati dalla FPGA nel modo seguente:

- $PKTEND\# = 1$;
- $SLOE\# = 0$;
- $SLRD\# = 1$;
- $SLCS\# = 0$;
- $SLWR\# = 1$;

- $A[0:1] = 3$;

e lo stream out rimane in questo stato per 2 colpi di clock perché l'FX3 campiona SLRD# due cicli di clock dopo che é passato dallo stato LOW a quello HIGH.

Infine, dopo un ciclo di clock, ritorna allo stato STREAM_OUT_IDLE in attesa della ricezione di altri dati o comandi da PC.

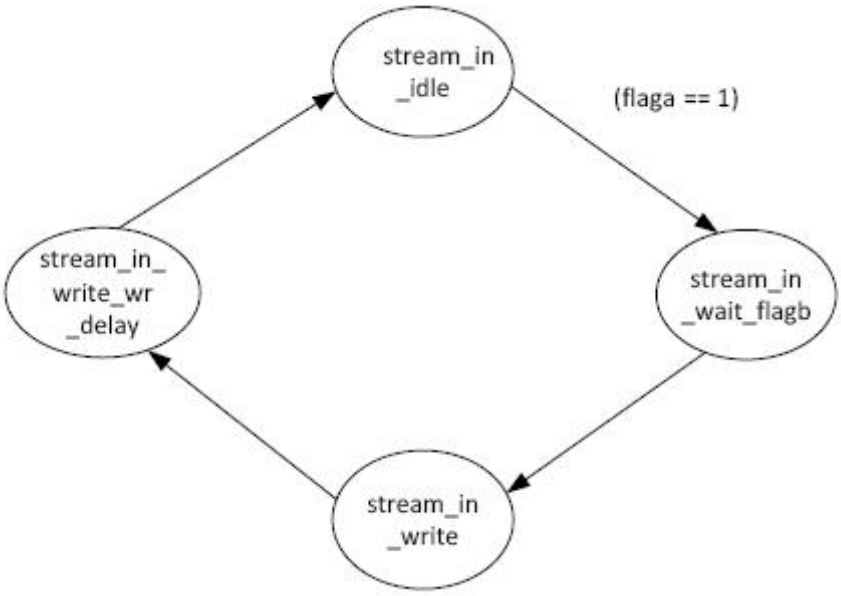

Figura 3.7: viene mostrata la macchina a stati dello STREAM IN stato per stato [56].

3.1.4 Stream in

Per poter capire come funziona lo stream in, facciamo riferimento allo schema a blocchi in figura 3.7 fornito dalla Cypress [56].

Lo STREAM IN parte dallo stato iniziale STREAM_IN_IDLE con i registri di comando asseriti in questo modo;

- $PKTEND\# = 1$;
- $SLOE\# = 1$;
- $SLRD\# = 1$;
- $SLCS\# = 0$;
- $SLWR\# = 1$;
- $A[0:1] = 0$;

Quando $FLAGA = 1$, lo STREAM IN passa allo stato STREAM_IN_WAIT_FLAGB. In qusto nuovo stato, passato un ciclo di clock, la mac-

china a stati passa allo stato successivo STREAM_IN_WRITE e l'FPGA setta i segnali di comando nel modo seguente:

- $PKTEND\# = 1$;
- $SLOE\# = 1$;
- $SLRD\# = 1$;
- $SLCS\# = 0$;
- $SLWR\# = 0$;
- $A[0:1] = 0$;

In questo modo l'FPGA puó inviare i dati all'FX3 tramite il bus a 16 bit. Una volta che il contatore di byte inviati alla FX3, fa passare il relativo flag dallo stato LOW a quello HIGH per indicare che il buffer si é riempito, termina la trasmissione dei dati e lo STREAM IN passa allo stato successivo: STREAM_IN_WRITE_WR_DELAY. In questo stato i segnale di controllo vengono settati in questo modo:

- $PKTEND\# = 1$;
- $SLOE\# = 1$;
- $SLRD\# = 1$;
- $SLCS\# = 0$;
- $SLWR\# = 1$;
- $A[0:1] = 0$;

e lo STREAM IN rimane in questo stato per 1 colpo di clock, poiché l'FX3 impiega un ciclo di clock a campionare il valore corretto di SLWR#.

Infine ritorna allo stato STREAM_IN_IDLE in attesa di trasmettere altri dati alla FX3.

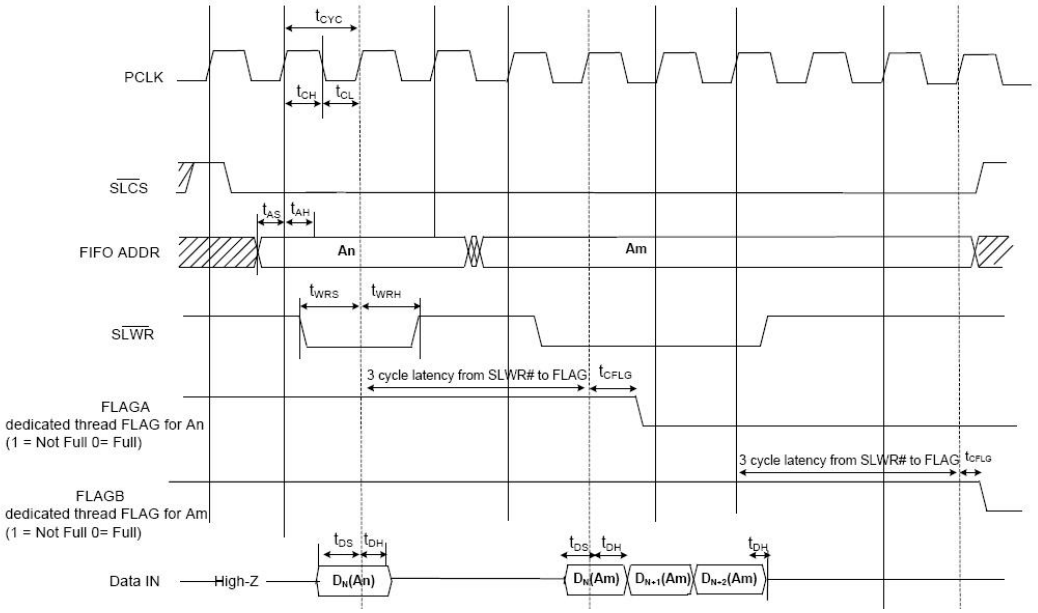

Figura 3.8: temporizzazioni dei segnali di STREAM IN [56], nel quale si osserva che il segnale di scrittura su FX3, (SLWR#), e i dati (DATA IN), devono essere precaricati sul fronte di discesa del clock, di modo che sul fronte di salita, quando la FX3 li campiona, essi siano aggiornati e stabili.

Dal momento che l'FX3 asserisce i flag sul fronte di salita del clock, la macchina a stati é stata sincronizzata su tale fronte. Poiché il datasheet dell'FX3 indica come tempo di setup 2 ns [45], abbiamo deciso di fornire questi segnali sul fronte di discesa in modo che siano stabili quando vengono campionati dall'FX3 sul successivo fronte di salita. Per far ció é stato creato un sottoprocesso dello STREAM IN, il quale viene attivato dal processo principale nel momento in cui si porta nello stato STREAM_IN_WRITE. Una volta attivo, svolge la sua attivitá sul fronte di discesa del clock e si occupa di gestire il segnale di scrittura su FX3, lo attiva nel momento in cui la FPGA deve trasferire dati alla FX3 e lo disattiva quando il buffer risulta pieno, inoltre precarica i dati sul bus per fare in modo che quando la FX3 li campiona, essi siano stabili. Nel momento in cui lo STREAM IN esce dallo stato STREAM_IN_WRITE, il sottoprocesso disabilita il segnale di scrittura e poi si mette in attesa di essere richiamato. Per sincronizzare poi la scrittura dei dati e dei comandi abbiamo scelto di utilizzare gli *ODDR*

(Dedicated Dual Data Rate Output Register) in figura 3.10 [40] . Questo componente viene comandato da 5 segnali:

- D1 e D2: sono i dati in ingresso;
- C: é il clock;
- CE: é il segnale (attivo HIGH) che abilita il componente;
- SR: é il segnale (attivo HIGH) che abilita il SET/RESET;

mentre in uscita fornisce:

- Q: segnale in uscita che si comporta come mostrato in figura 3.10.

  Questo componente é un registro che tiene in memoria i dati che gli giungono in ingresso, in particolare viene campionato sul fronte di salita del clock l'ingresso D1 e sul fronte di discesa D2 e i valori vengono poi inviati all'uscita Q sul fronte successivo. Nel nostro caso abbiamo utilizzato questo componente per i segnali che dovevano essere sincronizzati fra di loro, quindi per *SLWR#*, per i *dati* che vanno messi sul bus, e per il *clock* e ad ogni ODDR abbiamo fornito il clock negato dell'FPGA, dal momento che il processo per la trasmissione dati opera sul fronte di discesa. Questo ha permesso la sincronizzazione dei segnali fra FPGA e FX3.

3.1.5 Macchina a stati completa

Per il nostro progetto abbiamo la necessitá di unire insieme lo STREAM IN e lo STREAM OUT.

Vogliamo che il processo che parte da subito sia lo STREAM OUT e che questo sia sempre disponibile, perché é quello che permette di poter comandare tutti gli altri processi, in particolare di dare inizio allo STREAM IN per inviare i tempi delle misure o i risultati delle operazioni al PC. Lo STREAM OUT decodifica il comando inviato dal software e ne fa corrispondere un'operazione predefinita.

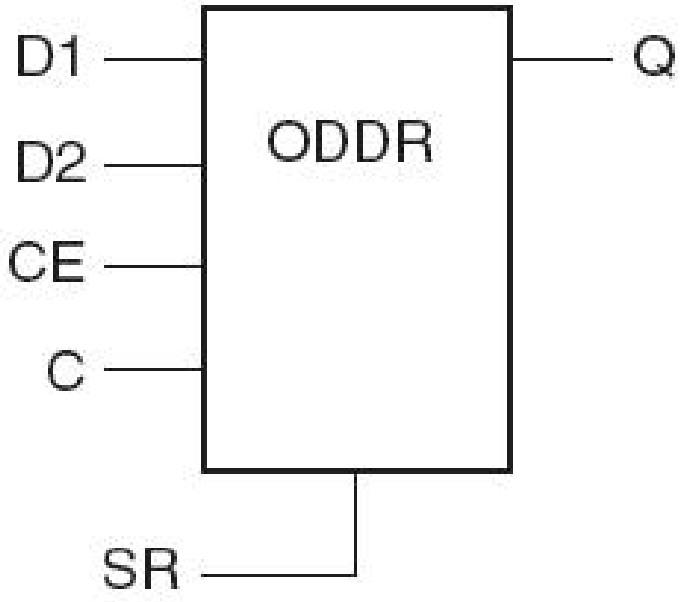

Figura 3.9: blocco che rappresenta l' ODDR dove si vedono i segnali di input (D1 e D2), il segnale di clock (C), il segnale di abilitazione (CE) e il segnale di set/reset(SR) [40].

Nel caso in cui venga attivato lo STREAM IN, si attiva un processo ausiliario che permette di passare dallo STREAM IN allo STREAM OUT e viceversa, in istanti scanditi da un contatore che é stato implementato nel VHDL. Abbiamo utilizzato questa modalitá di gestione della macchina a stati, perché come osservato con l'oscilloscopio andando a campionare il FLAG A (che indica quando il buffer della FX3 é disponibile, passando dallo stato LOW ad HIGH), abbiamo visto che la FX3 impiega circa 1 µs per passare, durante lo STREAM IN, da un buffer pieno ad un altro vuoto. Durante questa pausa il processo passa in modalitá STREAM OUT e controlla se sono stati inviati dei comandi. Se viene inviato il comando di stop, il download viene interrotto, altrimenti il processo, trascorsi 700 ns, ritorna in modalita STREAM IN. Queste operazioni vengono ripetute ciclicamente. In tale modo non viene portato via tempo allo STREAM IN, perché lo STREAM OUT viene eseguito in un tempo morto del processo di STREAM IN e quindi si ottimizza lo scaricamento dati da FPGA.

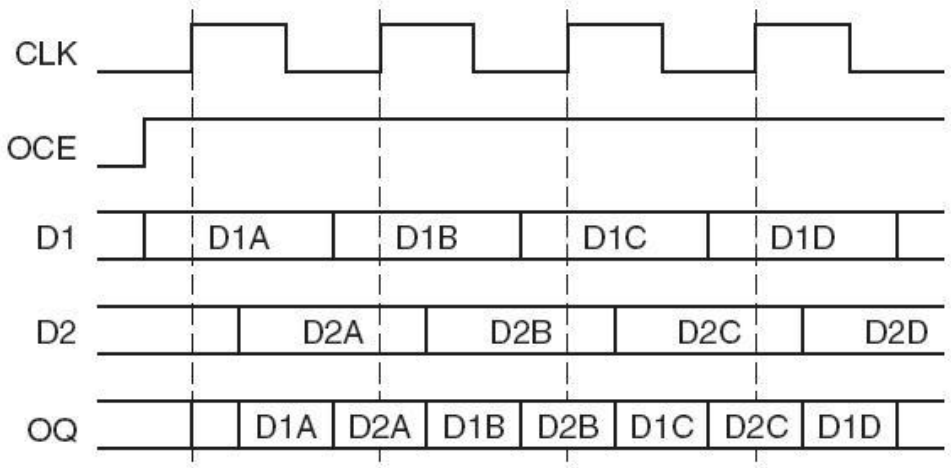

Figura 3.10: in ingresso viene campionato sul fronte di salita del clock l'ingresso D1 e sul fronte di discesa D2 e poi vengono inviati all'uscita Q in succesione e ció si ripete con i dati successivi [40].

3.2 L'interfaccia slave fifo

La EZ-USB FX3 viene programmata tramite un firmware fornito dalla casa costruttrice, lo SLAVE FIFO, che permette di configurare l'interfaccia per lo scambio di dati tra PC e FPGA tramite interfaccia USB 3.0. Per poter adattare ed ottimizzare il firmware al nostro dispositivo, abbiamo apportato modifiche, utilizzando il software fornito dalla casa costruttrice, in particolare al numero e alla dimensione dei buffer, seguendo le indicazioni di un data-sheet per l'ottimizzazione del throughput. Quindi alla fine delle modifiche, le caratteristiche principali del firmware, risultano essere;

- numerazione della EZ-USB FX3 con VID/PID 0x04B4/0x00F1;
- impostazione di FLAG A come full flag corrente;
- impostazione di FLAG B come partial flag con valore di watermark 2, dedicato al thread 0;
- impostazione di 2 buffer da 48 kbytes ciascuno per ottimizzare le prestazioni dello STREAM IN [57];
- impostazione di 1 buffer da 1 kB per lo STREAM OUT;

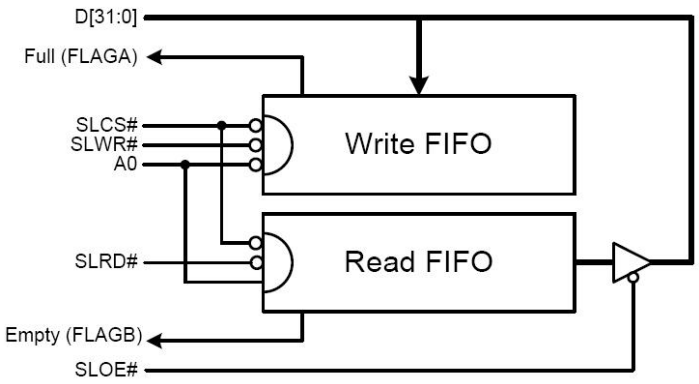

Figura 3.11: schema dell'interfaccia SLAVE FIFO con il blocco di scrittura e quello di lettura [43].

Il watermark serve ad impostare la soglia di attivazione del FLAG B e qunado é attivo indica che il buffer é quasi pieno a meno di un numero di parole (ogni parola corrisponde a 2 byte poiché trasmettiamo 16 bit alla volta) indicato nella eq 3.1:

$$numero\ di\ parole\ rimaste = watermark \cdot \frac{32}{numero\ di\ bit\ del\ bus\ dati} - 4 \tag{3.1}$$

nel nostro caso il bus dati é a 16 bit e il valore di watermark é 2 quindi le parole rimaste prima di riempire il buffer sono quelle dell'equazione (3.1) [56] ovvero zero.

Come detto in precedenza, utilizzare il FLAG B é risultato essere non accurato per cui abbiamo deciso di usare un contatore implementato nel codice VHDL per controllare il numero di parole presenti nel buffer.

3.3 La fase di test

Inizialmente il progetto della scheda era nato per ospitare un codice VHDL che permettesse di fare un cross-correlatore a 128 canali. Per motivi di tempo i cross-correlatori non erano pronti per essere testati sul nostro progetto.

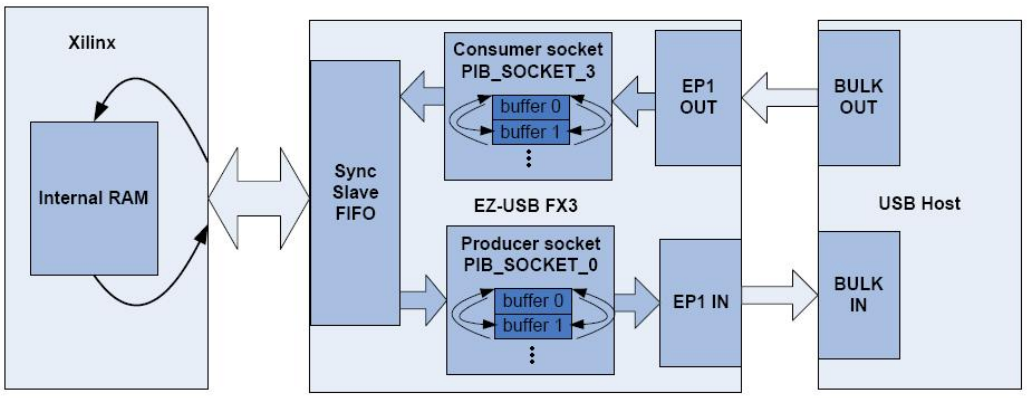

Figura 3.12: rappresentazione a blocchi della EZ-USB FX3 con buffer per lo STREAM IN e per lo STREAM OUT,interfaccia SLAVE FIFO e end point [56].

Abbiamo quindi deciso di passare ad implementare un codice VHDL che permettesse di fare time-stamping a 128 canali che verrá spiegato successivamente. Prima di implementare il time-stamping abbiamo deciso di ottimizzare il trasferimento dei dati via USB3.0. Per fare questo ci siamo avvalsi si una demoboard contenente una FPGA Spartan 6 (SP605) collegata alla demoboard tramite una scheda di ponte come si vede in figura 3.13. Poi sulla spartan 6 é stato caricato il codice VHDL della macchina a stati trattato nella sezione (3.1) e la FX3 é stata configurata per operare in modalitá slave fifo con le caratteristiche trattate nella sezione (3.2).

I test sono stati svolti con bus dati della FX3 a 32 bit in quanto sulla demoboard avevamo a disposizione il bus completo, mentre nel nostro progetto, abbiamo dovuto utilizzare un bus dati a 16 bit per i motivi spiegati nel capitolo 2.

3.3.1 Primo test

Come prima fase di test abbiamo creato un pattern di dati costanti per assicurarci che STREAM IN e STREAM OUT funzionassero. Per fare ció abbiamo inviato il comando di start STREAM IN tramite interfaccia PC. L'FPGA una volta ricevuto e decodificato il comando, ha inviato un pattern costante di dati come si vede in figura 3.14.

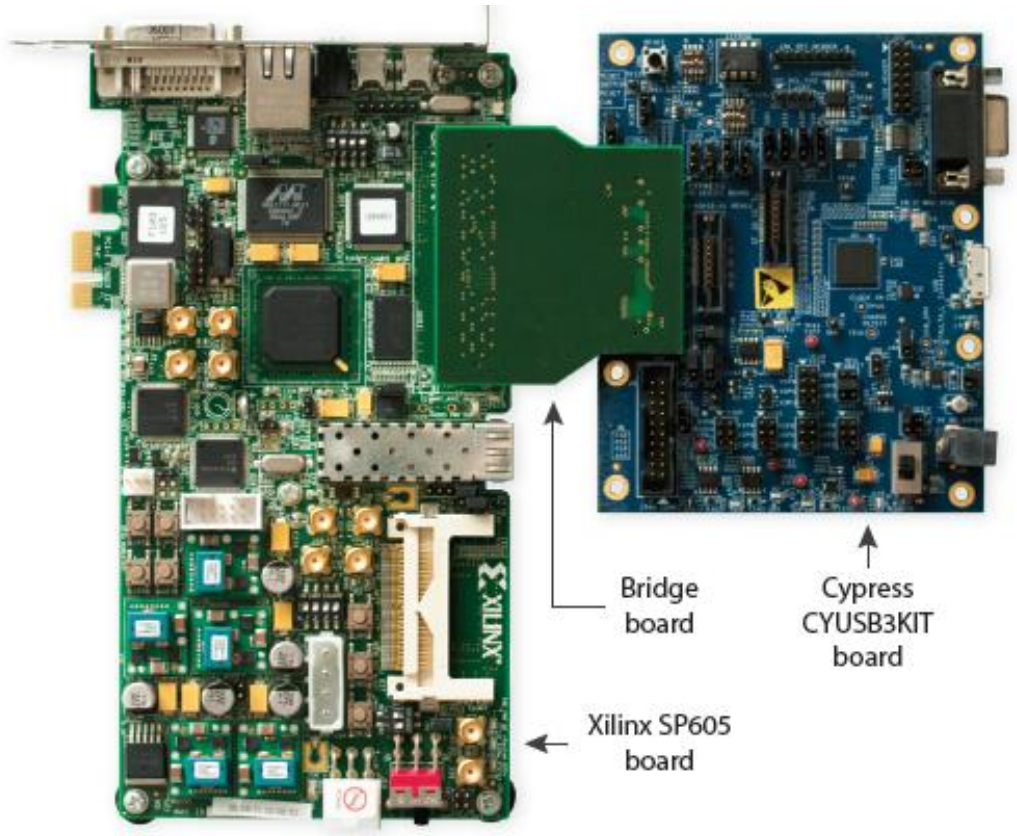

Figura 3.13: la Xilinx SP605 é utilizzata per generare un pattern conosciuto e spedirlo, alla Cypress CYUSB3KIT attraverso un bridge. Alla fine il dato é ricevuto sul PC e memorizzato in una SSD con un softweare C# o LABWIEV.

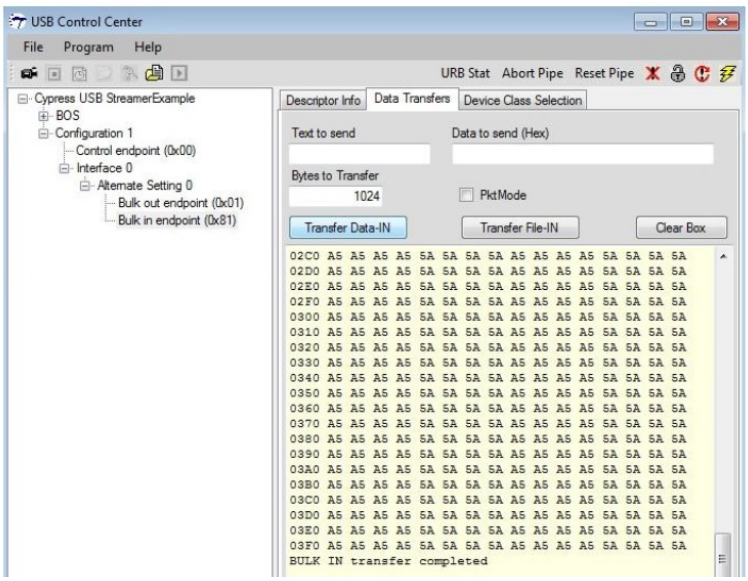

Figura 3.14: risultato del primo test nel quale si osserva un pattern di dati costante [43].

Una volta appurato che i dati ricevuti erano corretti, siamo passati alla seconda fase di test.

3.3.2 Secondo test

In un primo momento abbiamo testato la velocitá massima alla quale potevamo trasferire dati senza salvarli e per vedere ció abbiamo utilizzato un TOOL fornito dalla Cypress, *STREAMER*, che permette di vedere la velocitá di trasferiemto dati tra FPGA e FX3 senza il loro salvataggio come mostrato in figura 3.15.

Poiché siamo riusciti a raggiungere una velocitá di 360 MB/s (massimo teorico 400 MB/s), siamo passati alla seconda fase, nella quale durante lo STREAM IN trasferivamo una rampa di dati al PC e la salvavamo con un software sviluppato da noi in C# (che tratteremo nel capitolo 3.4), e analizzata.

Da questa analisi é risultato che salvando i dati il throughput si é abbassato a circa 350 MB/s Dal momento che non sono stati trovati errori nei

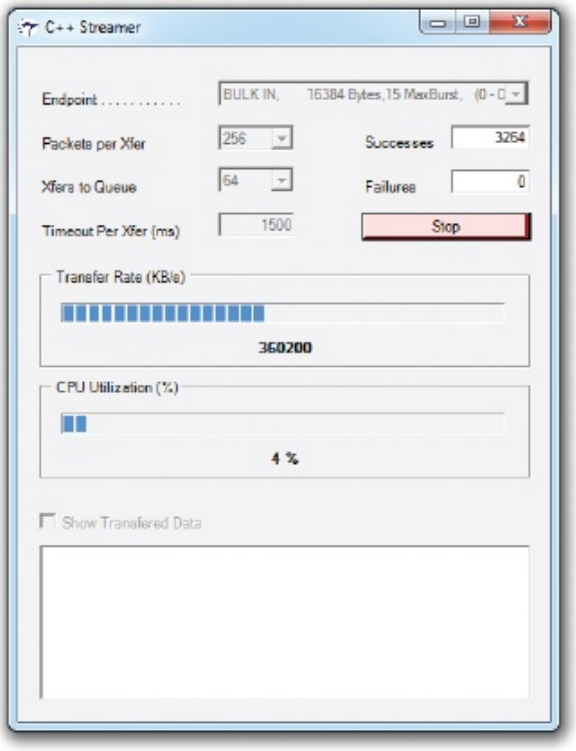

Figura 3.15: si puó vedere il tool della cypress, *STREAMER* [43], nel quale viene indicata la velocitá di trasferimento dati, senza il salvataggio.

3GB trasferiti, il sistema é stato validato e si é passati a sviluppare il sistema finale

3.4 Le interfacce grafiche

In questo capitolo vengono illustrate due interfacce grafiche, che svolgono la stessa funzione, ovvero quella di gestire da PC la macchina a stati e il processo di TIME-STAMPING e successivamente salvare i dati inviati da FPGA su SSD.

Una prima versione di interfaccia grafica che sarebbe servita per i cross-correlatori, era stata sviluppata in C#. A seguito dell utilizzo del nostro sistema per applicazioni di time-stamping, le quali ci sono state richieste dalla

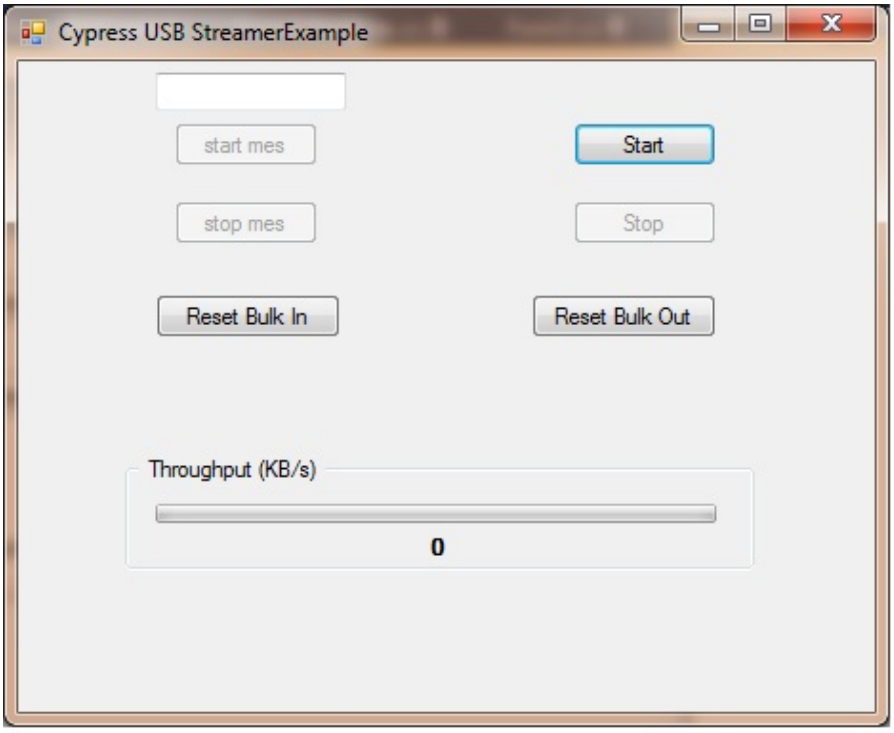

Figura 3.16: interfaccia grafica sviluppata in C# che permette di attivare o disattivare il processo di STREAM IN (START/STOP) e quello di TIME-STAMPING (START/STOP MEASURE).

University of California Los Angeles (UCLA), abbiamo deciso di sviluppare una interfaccia in LABVIEW, ambiente di lavoro di suddetta univeritá.

3.4.1 L'interfaccia in C#

Come si vede in figura 3.16: quest'interfaccia presenta 6 pulsanti:

- START: con questo tasto si attiva la macchina a stati dello STREAM IN e di conseguenza giungono dati dalla FPGA, questi vengono salvati in memoria;

- STOP: con questo tasto si disattiva la macchina a stati dello STREAM IN e non é possibile salvare dati;

- START MEASURE: con questo tasto si attiva il processo di TIME-STAMPING;
- STOP MEASURE: con questo tasto si disattiva il processo di TIME-STAMPING;
- RESET BULK IN: con questo tasto si svota il buffer della EZ-USB FX3 relativo allo STREAM IN;
- RESET BULK OUT: con questo tasto si svota il buffer della EZ-USB FX3 relativo allo STREAM OUT;

Oltre a questi pulsanti é presente una barra di progressione che visualizza la velocitá alla quale si sta effettuando il trasferimento dati fra FPGA e FX3.

Per una normale operazione di *TIME-STAMPING* i passi da seguire sono i seguenti:

1 START: cosí si attiva la macchina a stati che effettua lo STREAM IN dei dati da FPGA;

2 START MEASURE: cosí si attiva il processo di TIME-STAMPING e i dati nel frattempo vengono salvati nella memoria;

3 STOP MEASURE: per disattivare il TIME-STAMPING nel momento in cui si vuole terminare la fase di misura;

4 STOP: per disattivare la macchina a stati dello STREAM IN;

5 RESET BULK IN/RESET BULK OUT: per svuotare i buffer della FX3 e cosí preparare il tutto per una nuova misura;

3.4.2 L'interfaccia in LABVIEW

Per creare quest'interfaccia siamo partiti da quella sviluppata in linguaggio C#. Di quella precedente abbiamo preso le funzioni di nostro interesse, ovvero quelle che permettono di:

- controllare se il dispositivo EZ-USB FX3 é settato;

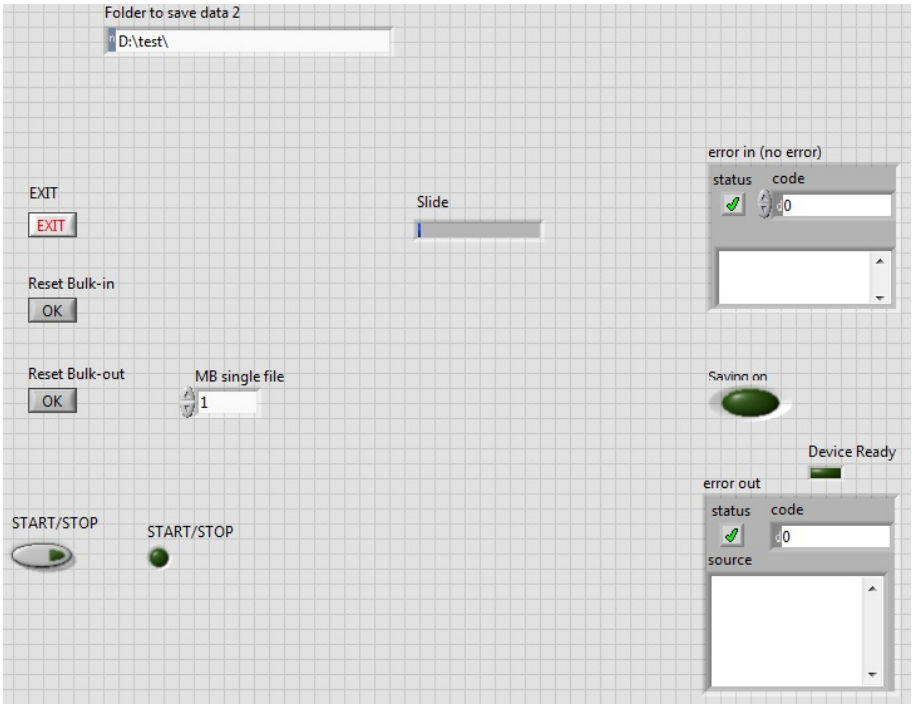

Figura 3.17: interfaccia grafica sviluppata in LABWIEV della quale si possono vedere i comandi principali come EXIT per uscire dall'interfaccia, START che in sequenza attiva la macchina a stati dello STREAM IN e il processo di TIME-STAMPING, STOP che in sequenza, disattiva il processo di misurazione e poi quello della macchina a stati dello STREAM IN e quelli di RESET BULK IN / BULK OUT.

- inviare comandi da PC alla FPGA;
- salvare i dati ricevuti su SSD;

Come in C#, in quest'interfaccia ci sono dei pulsanti principali:

- START: attiva in sequenza il processo di STREAM IN e di TIME-STAMPING e salva i dati ricevuti su SSD;
- STOP: disattiva in sequenza il processo di TIME-STAMPING e di STREAM IN e di conseguenza anche il salvataggio di dati;
- RESET BULK IN/ BULK OUT: svuota rispettivamente il buffer di STREAM IN e di STREAM OUT;

- EXIT: serve per uscire dall'interfaccia LABWIEV;

Le due schermate di ERROR IN e ERROR OUT sono utilizzate in fase di debug per controllare se i comandi che vengono spediti dall'interfaccia alla FX3 vengono ricevuti.

Per effettuare il processo di misura con questa interfaccia le fasi da seguire risultano:

1 nella sezione *FOLDER TO SAVE DATA* si inserisce il percorso della cartella nella quale si vogliono salvare i dati;

2 si preme START per attivare i processi di STREAM IN e TIME-STAMPING e quindi salvare i dati delle misure;

3 quando si vuole terminare la fase di misura si preme STOP per terminare il TIME STAMPING e lo STREAM IN;

4 premere RESET BULK IN / BULK OUT per preparare la FX3 ad una nuova fase di misura;

5 premere EXIT nel caso in cui si voglia uscire dall'interfaccia;

3.5 Il TIME-STAMPING

Con la tecnica *TIME-STAMPING* si misura il tempo di arrivo dei fotoni rispetto all'inizio dell'esperimento. Per sviluppare il firmware VHDL siamo partiti da una versione precedente sviluppata in un progetto in collaborazione con UCLA. In questo progetto é stata usata una FPGA Spartan 6 prodotta da Xilinx collegata ad un array 12x4 di SPAD. Nel nostro caso abbiamo adattato il tutto per poterlo utilizzare con una Kintex 7 di Xilinx. All'inizio abbiamo tenuto inalterati i 48 canali e poi in seguito, una volta che tutto funzionava, siamo passati a *128 canali* per poter collegare al nostro dispositivo due mante con array di spad di 8x8 ciascuna. Per ciascun evento vengono salvati 8 byte che sono organizzati nel modo seguente:

- 1 byte per indicizzare il canale in cui si é verificato l'evento ovvero l'arrivo di un fotone;
- 1 byte per segnalare errori (ad esempio per avvertire che la fifo del canale é quasi piena e che quindi si potrebbero perdere dati);
- 6 byte per il tempo di arrivo del singolo fotone;

Con *6 byte* per il tempo si puó arrivare a contare fino a:

$$t_{max} = 2^{nbit} \cdot T_{clock} = 2^{6byte \cdot 8bit} \cdot 10ns \approx 2,8 \cdot 10^6 s \approx 32giorni \tag{3.2}$$

mentre con il progetto originario venivano usati solo 3 byte per il tempo di arrivo, che quindi al massimo poteva essere pari a *167 ms*. Dato il fondo scala ridotto si utilizzava un contatore ausiliario di *ROLLOVER* che contava quante volte era stato raggiunto il tempo massimo di 167 ms. L'enorme vantaggio portato dall'adozione di un chip USB3.0 permette di ottenere un fondo scala di misura nettamente superiore e di aumentare il numero di canali a disposizione.

Come si puó vedere dalla figura 3.18 il processo di TIME-STAMPING utilizza due tipi di FIFO:

- s-FIFO (1 KB): una per canale che memorizza le parole da 8 byte del relativo canale;
- b-FIFO(2KB): questa é una sola e svolge la funzione di tramite tra le s-FIFO e la macchina a stati dello STREAM IN alla quale fornisce i dati da inviare al PC. Inoltre permette di rendere asincroni scrittura dati al suo interno e lettura degli stessi, nel caso in cui si lavorasse con frequenze diverse;

Il funzionamento del time-stamping ha richiesto test preliminari con lo scopo di verificare il funzionamento di ogni singolo passaggio dello schema di time-stamping. Ovvero abbiamo testato il salvataggio di dati generati in ingresso all'interno di ciascuna s-FIFO. Successivamente si é verificato il

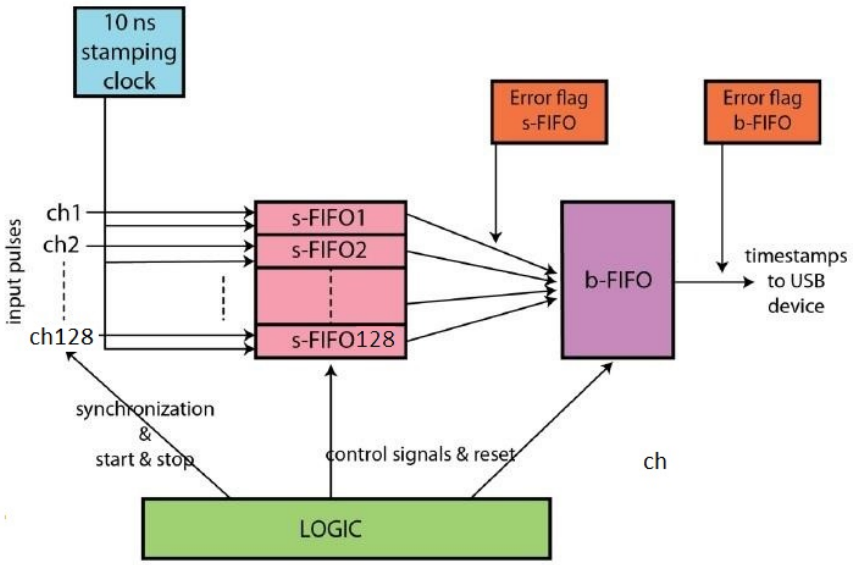

Figura 3.18: Schema a blocca del TIME-STAMPING: a sinistra ci sono le s-FIFO che memorizzano gli 8 byte del singolo evento relativo all'arrivo del fotone all'ingresso del relativo canale e a destra c'é la b-FIFO che fa da ponte fra le s-fifo e la macchina a stati dello STREAM IN alla quale fornisce i dati da inviare al PC.

salvataggio di questi dati da s-FIFO a b-FIFO e solo a questo punto abbiamo verificato lo scaricamento dati verso PC verificandone la loro correttezza.

Per effettuare le misure di TIME-STAMPING, per prima cosa tramite PC si deve inviare il comando che attiva lo STREAM IN. Successivamente si fornisce il comando che attiva la fase di misura e questo comporta l'attivazione di un contatore a 6 byte che viene incrementato ogni 10 ns. Una volta cominciata la fase di misura ogni canale salva nella rispettiva s-FIFO parole da 8 byte ogni volta che lo SPAD rivela un fotone nella s-FIFO. Ogni s-FIFO viene interrogata ed il suo contenuto trasferito alla b-FIFO. Il tutto si ripete per ogni canale ciclicamente. La b-FIFO fornisce poi i dati di STREAM IN per poter essere inviati alla FX3 e successivamente al PC. Poiché i singoli dati sono da 8 byte, e il bus dati che li trasferisce alla FX3 é da 2 byte (16 bit), il dato viene spezzato in quattro parti che verranno trasmesse una alla volta.

Capitolo 4

Risultati sperimentali

In questo capitolo vogliamo presentare i risultati ottenuti che hanno come scopo quello di verificare il corretto funzionamento dei firmware e software descritti nel Cap.3 per applicazioni di time-stamping.

4.1 Test del time-stamping

Per la fase di test sono stati implementati nel codice VHDL piú contatori che generano onde quadre sui 128 canali a varie frequenze nell'intervallo 1kHz-125kHz. Questo test serve a simulare la presenza di ingressi veri, rappresentati in futuro da array di SPAD. Usando frequenze note il riscontro sulla correttezza dei risultati é immediato. Il fronte di salita dell'onda quadra é stato considerato come l'evento di rilevamento del fotone da parte del singolo SPAD. Dal momento che gli eventi sono periodici, se si analizzano i tempi di arrivo all'interno di un unico canale é necessario che questi siano equispaziati. Detta $\Delta conteggi$ la differenza fra 2 tempi di arrivo adiacenti si ottiene che:

$$T_{onda\,quadra} = \Delta conteggi \cdot 10ns \tag{4.1}$$

Le figure 4.1 e 4.2 mostrano in ascissa il numero di eventi cioé l'arrivo degli impulsi e in ordinata i $\Delta conteggi$ ovvero il rapporto tra il periodo dell'onda quadra generata del canale considerato, e il periodo di campionamento

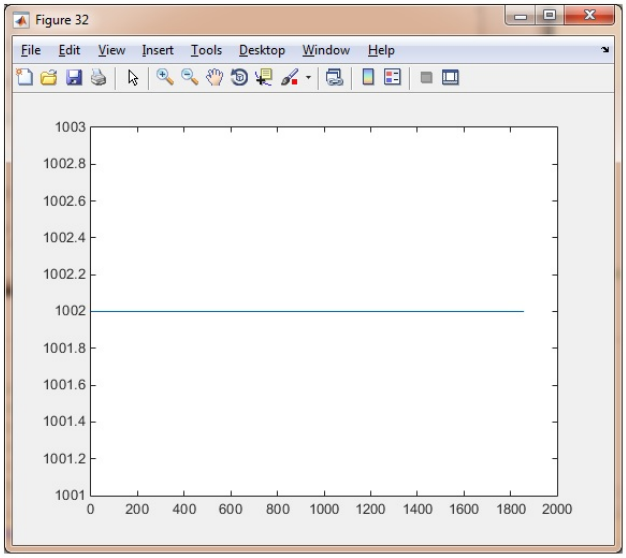

Figura 4.1: grafico di Matlab che rappresenta i conteggi per l'onda quadra di test relativa al canale a 100kHz.

dei dati pari a 10nS in quanto nella macchina a stati si utilizza un clock a 100MHz. Si nota che, nonostante si stiano analizzando i canali piú veloci

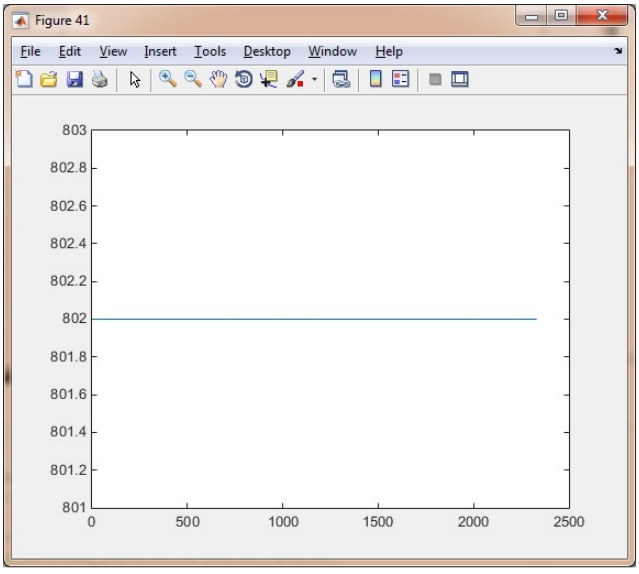

Figura 4.2: grafico di Matlab che rappresenta i conteggi per l'onda quadra di test relativa al canale a 125kHz.

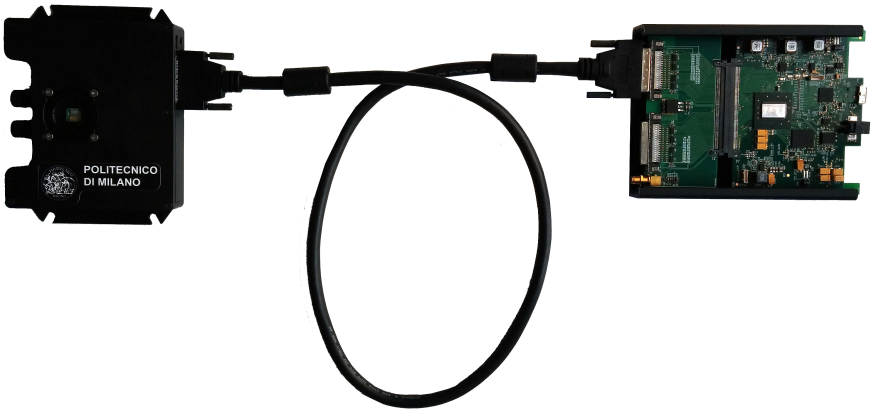

Figura 4.3: Setup utilizzato per testare i conteggi. L'array di SPAD risulta essere coperto in quanto si vuole fare una misura di buio.

il rapporto rimane sempre costante poiché questa frequenza é comunque inferiore alla massima frequenza a cui puó funzionare ogni canale che risulta essere pari a:

$$F_{max} = \frac{1}{4 \cdot 10\,\text{ns} \cdot 128} \approx 195\,\text{kHz} \tag{4.2}$$

Con 10ns tempo in cui la macchina a stati interroga le s-FIFO per vedere se contengono dati, 4 sono i cicli di clock impiegati dalla macchina a stati per scaricare il contenuto di una s-FIFO e 128 é il numero di s-FIFO che abbiamo allocato nella macchina a stati.

4.2 Test di buio

Il setup utilizzato per testare il conteggio medio dei test di buio prevede l'utilizzo di una manta da 32 canali collegata tramite cavo SCSI al nostro sistema come rappresentato in figura 4.3.

Canale	A	B	Canale	A	B
1	334	403	17	10285	10228
2	464	430	18	1000	969
3	846	813	19	563	563
4	1445	1540	20	7198	7360
5	73572	71352	21	902	849
6	1312	1483	22	7841	7624
7	574	560	23	8107	7871
8	5018	5232	24	15369	15460
9	8844	8694	25	661	766
10	645	602	26	597	467
11	548	545	27	483	524
12	8108	8150	28	550	605
13	12148	12476	29	5654	5508
14	3571	3468	30	7152	6916
15	638	623	31	685	652
16	5144	5018	32	626	567

Tabella 4.1: Tabella comparativa tra i due conteggi di buio ottenuti mediante due metodi di analisi differenti.

Scopo di questo test é quello di verificare che il numero di conteggi medio al secondo per ogni canale (colonna A) sia simile a quello che si ottiene eseguendo lo stesso test ma analizzando i dati mediante software sviluppato all' interno del nostro gruppo di ricerca (colonna B). Usando la nostra scheda e un software C#, é stato effettuato uno scaricamento dati e successivamente, tramite software Matlab sono stati estratti il numero di conteggi di fotoni per ciscun canale registrati all' interno della finestra temporale di campionamento pari a 10ns. La tabella 4.1 evidenzia che i risultati ottenuti usando i due metodi (A e B) sono del tutto soddisfacenti in quanto non vi é un eccessivo scarto.

Considerando ora che il tempo morto dei circuiti di quenching degli AQC degli SPAD della manta é pari a 150ns, abbiamo verificato che questo tempo, normalizzato al periodo di campionamento (10ns) che l'elettronica impiega per verificare se é presente un fotone, sia inferiore a 15. Per fare questo abbiamo analizzato tramite Matlab il file ottenuto dal precedente test di buio. Dalla figura 4.4 si osserva che, anche considerando differenze temporali

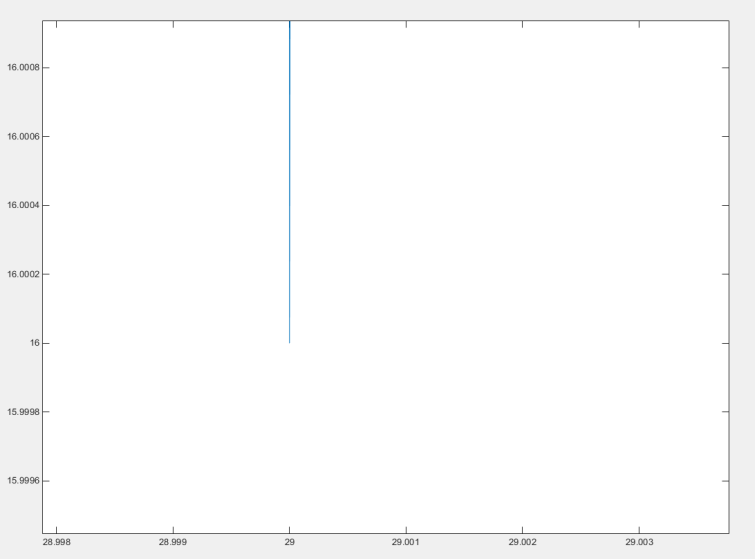

Figura 4.4: Dettaglio del grafico Matlab che riporta in ascissa il numero di eventi e in ordinata la differenza tra l'arrivo di due fotoni consecutivi normalizzata al periodo di campionamento.

brevi tra l'arrivo di due fotoni consecutivi, essa si mantiene al di sopra di 15.

4.3 Time-stamping

Il setup prevede il collegamento di un generatore di impulsi al connettore SMA della nostra scheda come mostrato in figura 4.5

L'esperimento consiste nel generare un numero noto (9) di pacchetti di impulsi (9000) ad una data frequenza (322kHz), analizzando il file salvato usando un sofrware C# tramite un programma Matlab, verificare che il numero di pacchetti di impulsi ricevuto sia pari a quello generato e che la frequenza di tali impulsi sia la medesima con quella scelta.

Dal grafico in figura 4.6 si puó notare la presenza di picchi di ampiezza proporzionale all' intervallo di tempo che intercorre tra la generazione di un pacchetto di impulsi e il successivo. Infatti, finito il pacchetto di impulsi il software effettua la diferenza tra il successivo pacchetto che peró arriva solo dopo un certo intervallo di tempo pari al tempo che intercorre tra la successiva pressione del bottone che genera un nuovo pacchetto di impulsi.

Figura 4.5: Figura che mostra il setup utilizzato per l'esperimento. Un generatore di impulsi collegato alla scheda.

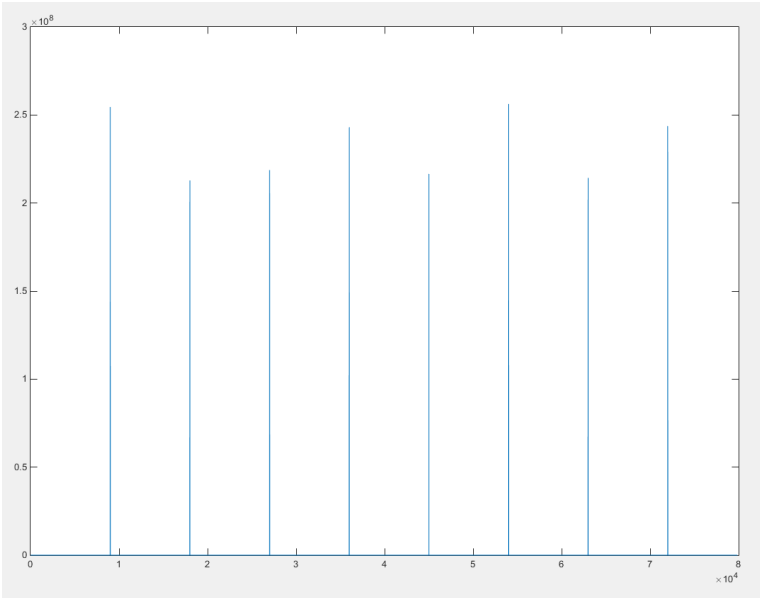

Figura 4.6: Grafico Matlab mostra in ascissa 9 pacchetti da 9000 impulsi ciascuno e in ordinata la differenza di arrivo degli impulsi normalizzata al periodo di campionamento degli stessi.

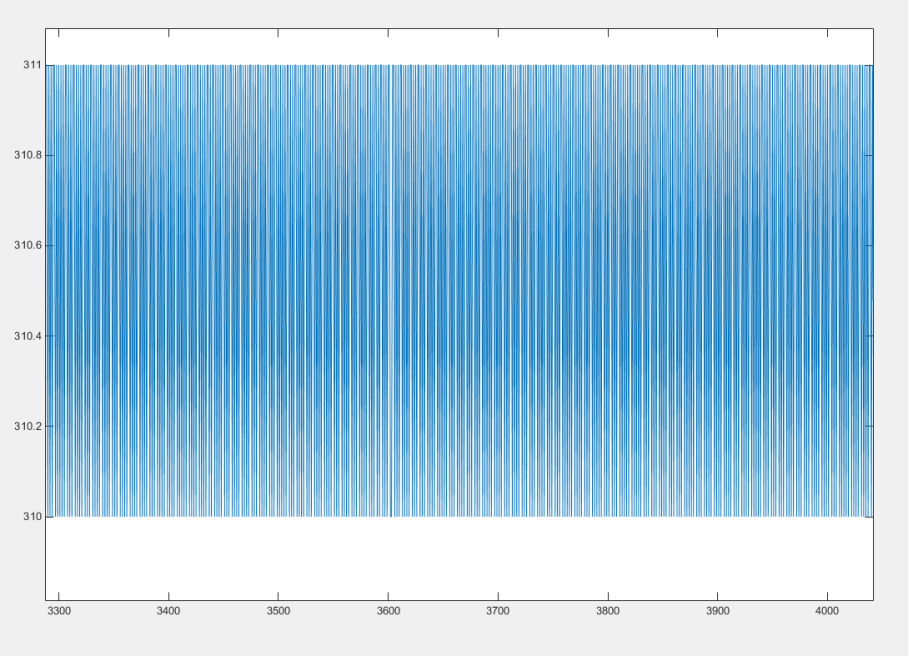

Figura 4.7: Dettaglio dell' andamento dei pacchetti di impulsi.

Volendo analizzare meglio cosa accade tra un impulso e il successivo si osserva che in ordinata non é presente una linea continua ma un barba di valori come mostrato in figura 4.7.

Questo é dovuto alla differenza seppur minima del clock con cui vengono generati il gli impulsi da parte del generatore e quella del clock interno alla macchina a stati dell FPGA che legge tali impulsi. Il rapporto tra il clock con cui si campionano gli impulsi (100MHz) il clock con cui vengono generati gli impulsi (322kHz) rimane comunque entro una fascia di valori accettabili ovvero non si osservano picchi dovuti a possibili errori di campionamento degli impulsi ricevuti a conferma della bontá della macchina a stati interna all'FPGA che effettua il time-stamping.

Capitolo 5

Conclusioni e sviluppi futuri

Durante il nostro lavoro di tesi é stato realizzato un sistema compatto che presenta due ingressi a cui si collegano le mante. Il sistema completo é rappresentato in figura 5.1

La motivazione che ha portato allo sviluppo di questo sistema risulta in una maggiore flessibilitá data dall' avere una sola FPGA e due ingressi invece che usare 2 FPGA separate e gestire il flusso di dati risultanti dalle operazioni di timestamping. Inoltre le FPGA precedentemente in possesso nel gruppo di ricerca presentano presentazioni sicuramente inferiori e una velocitá di trasferimento dati limitata dall'utilizzo di un protocollo USB2.0. L'uscita con cui vengono scaricati dati verso il PC é affidata ad una connessione veloce USB3.0. L'impiego primario per cui é stato progettato questo sistema riguarda applicazioni che sfruttano muduli SPAD in cui si richiedono misure di tipo FCS e FRET per misure da laboratorio. Abbiamo inoltre creato una interfaccia C# e Labview in grado di comunicare col sistema. La versatilitá data dalla presenza di una scheda connettori facilmente ridisegnabile per soddisfare le piú svariate esigenze e la presenza di un FPGA con buone capacitá computazionali rendono questo sistema facilmente riprogrammabile e in grado di soddisfare specifiche elevate.

Come accennato in precedenza in questa tesi non siamo riusciti a sfruttare a pieno le potenzialitá dell' FX3 in quanto a causa di un errore di layout siamo stati in grado di usare solo 16 delle 32 linee di bus a disposizione. In

Figura 5.1: Sistema completo posto all'interno della sua scatola. Nella foto é stato posto un coperchio di vetro per meglio visualizzare le due schede al suo interno.

particolare questo errore é scaturito da una inversione di collegamento tra una linea dati e una linea di controllo. Nonostante questo inconveniente nel layout, siamo riusciti ad ottenere risultati piú che soddisfacenti riuscendo a rispettare le specifiche di progetto ovvero effettuare time-stamping su 128 canali con ingressi a frequenze medie di 100kHz per verificare le specifiche del progetto che richiedevano un conteggio di 100 kcps di media su tutti i canali. L' idea iniziale era quella di sviluppare un sistema da collegare a matrici da 64 SPAD. I test sono stati svolti con matrici di 32 SPAD poiché allo stato attuale le prime sono ancora in fase di sviluppo. Mentre i test con frequenze fisse generate all'interno del software VHDL sono stati fatti su tutti i 128 canali. Versioni successive di questa scheda permetteranno di raggiungere la piena velocitá teorica della FX3 permettendo allo stesso tempo di aumentare le potenzialitá di utilizzo. La presenza di una memoria SRAM, anche se non usata nel nostro progetto, verrá usata in seguito non appena saranno terminati dei test che vogliono accertarne il corretto funzionamento. Questo consentirá alla nostra scheda di essere utilizzata anche come demo-board all' interno del laboratorio per testare progetti futuri.

Bibliografia

[1] Haupts U Maiti S. and WebbWW. Fluorescence correlation spectroscopy: diagnostics for sparse molecules. *Proc Natl Acad Sci U.S.A.*, Oct. 1997. 3

[2] Haustein E. and Schwille P. Fluorescence correlation spectroscopy: novel variations of an established technique. *Annu Rev Biophys Biomol Struct.*, 2007. 3

[3] Heikal AA Hess ST, Huang S and Webb WW. Biological and chemical applications of fluorescence correlation spectroscopy: a review. *Biochemistry*, Jan. 2002. 3

[4] Thompson N L. *Topics in Fluorescence Spectroscopy Techniques.* Plenum Press, New York, 1991. 3

[5] Oehlenschläger F. Walter NG Schwille P. Quantitative hybridization kinetics of dna probes to rna in solution followed by diffusional fluorescence correlation analysis. *Biochemistry*, Aug. 1996. 3

[6] Trier U Schäfer-Korting M Schüler J, Frank J and SaengerW. Interaction kinetics of tetramethylrhodamine transferrin with human transferrin receptor studied by fluorescence correlation spectroscopy. *Biochemistry*, Jan. 1999. 3

[7] Hovius R Wohland T, Friedrich K and Vogel H. Study of ligand-receptor interactions by fluorescence correlation spectroscopy with different fluorophores: evidence that the homopentameric 5-hydroxytryptamine type 3as receptor binds only one ligand. *Biochemistry*, Jul. 1999. 3

[8] Causa F De Santo I and Netti PA. Mapping receptor density on live cells by using fluorescence correlation spectroscopy. *Analytical Chemistry*, Feb. 2010. 3

[9] Patra D Enderlein J, Gregor I and Fitter J. Art and artefacts of fluorescence correlation spectroscopy. *Curr Pharm Biotechnol*, Apr. 2004. 3

[10] Nishimura G and Kinjo M. Systematic error in fluorescence correlation measurements identified by a simple saturation model of fluorescence. *Analytical Chemistry*, Apr. 2004. 3

[11] Lee J Davis LM Deininger P Lee W, Lee YI and Soper SA. Cross-talk-free dualcolor fluorescence cross-correlation spectroscopy for the study of enzyme activity. *Analytical Chemistry*, Feb. 2010. 3

[12] ChengWW Stults DA Wiebracht ER Kasianowicz JJ Culbertson MJ, Williams JT and Burden DL. Numerical fluorescence correlation spectroscopy for the analysis of molecular dynamics under nonstandard conditions. *Analytical Chemistry*, Jun. 2007. 3

[13] Benda A Hof M Kapusta P, Wahl M and Enderlein J. Fluorescence lifetime correlation spectroscopy. *Journal of Fluorescence*, Jan. 2007. 3

[14] Dominik Wöll. Fluorescence correlation spectroscopy in polymer science. *RSC Advances*, 2014. 4

[15] D. Hellweg T. Nielsen, M. Frick and P. Andresen. High efficiency beam splitter for multifocal multiphoton microscopy. *J. Microsc*, Mar. 2001. 6

[16] Keiichiro Kagawa Yusuke Ogura and Jun Tanida. Optical manipulation of microscopic objects by means of vertical-cavity surface-emitting laser array sources. *Applied Optics*, 2001. 6

[17] Rech I Gulinatti A Ghioni M Cova S Weiss S Colyer RA, Scalia G and Michalet X. High-throughput fcs using an lcos spatial light modulator and an 8x1 spad array. *Biomedical Optics Express*, 2010. 6

[18] G. Scalia A. Ingargiola R. Lin J. E. Millaud S. Weiss O.H. W. Siegmund A.S. Tremsin J.V. Vallerga A. Cheng D. Aharoni M. Levi K. Arisaka F. Villa F. Guerrieri F. Panzeri I. Rech A. Gulinatti F. Zappa M. Ghioni X. Michalet, R. A. Colyer and S. Cova. Development of new photon-counting detectors for single-molecule fluorescence microscopy. *Philos Trans R Soc Lond B Biol Sci.*, 2012. 6

[19] L.Wallerman S.Wennmalm, H. Blom and R. Rigler. Uv fluorescence correlation spectroscopy of 2-aminopurine. *Biological Chemistry*, 2001. 6

[20] L.M. Davis A. Spinelli and H. Dautet. Actively quenched single-photon avalanche diode for high repetition rate time-gated photon counting. *Rev. Sci. Instrum.*, 1996. 6

[21] Martin Drewel Klaus Schatzel and sven Stimac. Photon correlation measurements at large lag times: improving statistical accuracy. *J. Mod optic*, 1988. 7

[22] Alec Sandy Brian Tieman, Suresh Narayanan and Marcin Sikorski. Mpi-correlator: a parallel code for performing time correlations. *Nucl. Inst. Meth.*, Sep. 2011. 9

[23] Davide Magatti and Fabio Ferri. Fast multi-tau real-time software correlator for dynamic light scattering. *Applied optics*, Aug. 2001. 11

[24] Weiss S. Fluorescence spectroscopy of single biomolecules. *Science*, 1999. 13

[25] Weiss S. Measuring conformational dynamics of biomolecules by single molecule fluorescence spectroscopy. *Nat. Struct. Mol. Biol*, 2000. 13

[26] Weiss S Michalet X and Jäger M. Single-molecule fluorescence studies of protein folding and conformational dynamics. *Chem. Rev.*, 2006. 13

[27] Ho S.O Kortkhonjia E Weiss S Kapanidis A.N, Margeat E and Ebright R.H. Initial transcription by rna polymerase proceeds through dna-scrunching mechanism. *Science*, 2006. 13

[28] Laurence T.A Doose S Ho S.O. Kapanidis A.N, Margeat E. Retention of transcription initiation factor sigma70 in transcription elongation: single-molecule analysis. *Mol. Cell*, 2005. 13

[29] Ebright Y.W Korlann Y Kortkhonjia E. Chakraborty A, Wang D. Opening and closing of the bacterial rna polymerase clamp. *Science*. 13

[30] Davis L.M. Li L.Q. Single-photon avalanche-diode for single-molecule detection. *Rev. Sci. Instrum.*, 1993. 13

[31] Rech I Gulinatti A. Coyler R.A, Scalia G. High-throughput fcs using an lcos spatial light modulator and an 8x1 spad array. *Biome Optics Express*, 2010. 13

[32] Kim T Rech I Resnati D. Coyler R.A, Scalia G. High-throughput multispot single-molecule spectroscopy. *Proc of SPIE 7571*, 2010. 13

[33] Sarkosh N Gulinatti A. Ingargiola A, Panzeri F. 8-spot smfret analysisusing two 8-pixel spad array. *Proc of SPIE 8590*, 2013. 13

[34] Nam Ki Lee E. Margeat X. Kong A. N. Kapanidis, T. A. Laurence and S. Weiss. Alternating-laser excitation of single molecules. *Accounts of Chemical Research*, 2005. 18

[35] Robert H. Hadfield. Single-photon detectors for optical quantum information applications. *Nature Photonics*, 2009. 20

[36] S. Marangoni. Ultra-compact electronic systems for luminescence analysis with single photon detectors. *PhD thesis*, 2009. 23

[37] Fairchild Semiconductor. *FXLA104 datasheet.* http://www.fairchildsemi.com/datasheets/FXLA104.pdf. 31

[38] Analog Devices. *ADCMP606/ADCMP607.* http://www.analog.com/media/en/tecnical-documentation/data-sheets/ADCMP606_607.pdf. 31

[39] Xilinx. *7Series FPGAs Package & Pinout.* http://www.xilinx.com/support/documentation/user_guides/ug475_7Series_Pkg_Pinout.pdf. 35

[40] Xilinx. *7Series FPGAs SelectIO Resources.* http://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SlectIO.pdf. x, 35, 36, 78, 79, 80

[41] Xilinx. *7Series FPGAs Configurations.* http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf. 37, 44

[42] Cypress Semiconductor. *AN75705 datasheet.* www.cypress.com/?docID=46748. viii, x, 37, 38, 39, 67, 68, 69, 70

[43] Cypress Semiconductor. *AN87216 datasheet.* www.cypress.com/?rlID=84236. x, xi, 37, 81, 84, 85

[44] Cypress Semiconductor. *AN76405 datasheet.* www.cypress.com/?docID=49862. viii, 38

[45] Cypress Semiconductor. *CYUSB3014 datasheet.* www.cypress.com/?docID=50647. xiii, 39, 54, 77

[46] Cypress Semiconductor. *AN70707 datasheet.* www.cypress.com/?docID=50685. 39, 40

[47] Micron. *Micron Serial NOR Flash Memory.* http://www.micron.com/ /media/ Documents/Products/Data%20Sheet/NOR%20Flash/Serial%20NOR/N25Q/n25q_128mb_1_8v_65nm.pdf. 44

[48] Maxim Integrated. *MAX4751/MAX4752/MAX4753.* http://datasheets.maximintegrated.com/en/ds/MAX4751-MAX4753.pdf. viii, 44, 45

[49] Cypress Semiconductor. *CYC1425KV18 datasheet.* http://www.cypress.com/?docID=40501. viii, 47, 48, 55

[50] Xilinx. *Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics.* http://www.xilinx.com/support/documentation/data_sheets/ds182_Kintex_7_Data_Sheet.pdf. xiii, 51, 52

[51] General Electric. *PVX012A0X datasheet.* http://apps.geindustrial.com/publibrary/checkout/PVX012A0X?TNR=Data%20Sheets%7CPVX012A0X%7Cgeneric. 52

[52] General Electric. *PVX006A0X datasheet.* http://apps.geindustrial.com/publibrary/checkout/PVX006A0X?TNR=Data 52

[53] Altera. *100G Development Kit, Stratix V GX Edition Reference Manual.* http://www.altera.com/literature/manual/rm_svgx_100g_dev_board.pdf. 56

[54] Vishay. *SiC413 datasheet.* http://www.vishay.com/docs/69057/sic413.pdf. 56

[55] Altera. *Power Distribution Network Design Tool.* http://www.altera.com/technology/signal/power-distribution-network/sgl-pdn.html. 58

[56] Cypress Semiconductor. *AN65974 datasheet.* http://www.cypress.com/?docID=47020. x, xi, 65, 66, 72, 75, 77, 81, 82

[57] Cypress Semiconductor. *AN86947 datasheet.* www.cypress.com/?docID=48135. 80