POLITECNICO DI MILANO
FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Specialistica in Ingegneria Elettronica

Implementazione di un sistema di acquisizione ed elaborazione digitale con logica configurabile per applicazioni di imaging e spettroscopia gamma

Relatore: Prof. Angelo Geraci
Correlatore: Ing. Andrea Abba

Tesi di Laurea di:
Alberto Cusimano
Matr. 739955

Anno Accademico 2010-2011

Indice

Indice delle Figure

Abstract

Questo lavoro di laurea si inquadra all'interno di un ampio progetto di ricerca finalizzato alla realizzazione di un sistema di rivelazione di eventi gamma a basso rate in un range energetico esteso. L'intero progetto è patrocinato dall'ESA (European Space Agency) ed ha come scopo lo sviluppo di uno spettrometro per osservatori di astrofisica su satellite.
L'intero sistema è composto dalle seguenti parti fondamentali: l'apparato di rivelazione, il circuito ASIC di front-end e la scheda di acquisizione digitale (DAQ).
Il sensore è formato da rivelatori a deriva allo stato solido (SDD – Silicon Drift Detectors) con risoluzione millimetrica. Il sistema di rilevazione è costituito da uno scintillatore Bromuro di Lantanio (LaBr3) il quale è accoppiato ad una matrice di rivelatori. La matrice è suddivisa in 9 quadranti da 9 rivelatori ciascuno. Il front-end è affidato a 3 ASICs, collegati a 3 quadranti ciascuno, i quali misurano l'energia della radiazione incidente su ogni SDD del detector. A fronte di un evento in arrivo sul sensore un sistema di trigger fotografa il valore energetico misurato da ciascuno degli 81 SDD. Le uscite degli ASICs vengono inviate alla scheda di acquisizione ed elaborazione l'uscita della quale deve essere inviata al PC host del sistema tramite interfaccia ethernet.
Le attività principali del mio lavoro hanno riguardato tutti gli aspetti hardware e firmware della scheda di acquisizione digitale.
Ho realizzato il firmware VHDL per il dispositivo FPGA che si occupa di tre funzioni principali. Generare i segnali di controllo per gli ASIC, controllare gli ADC raccoglierne i dati e inviare i dati acquisiti al microprocessore (un AP700 della ATMEL).
Il driver del microprocessore organizza i dati in arrivo dall'FPGA in pacchetti e li invia al PC via Ethernet.
La comunicazione Ethernet che ho implementato è quella standard ma con alcune specifiche particolari che hanno richiesto da parte mia un design custom dell'architettura firmware. In particolare sono stati posti come vincolanti le seguenti specifiche: 1) garanzia di alta velocità di trasferimento "error-free" fino a 150 kframes/sec; 2) elevata affidabilità tramite utilizzo di protocolli "safe" (TCP/IP); 3) architettura modulare con disaccoppiamento fisico scheda-sistema di acquisizione.
Al fine di sfruttare a pieno le potenzialità della rete Ethernet, di è optato per l'utilizzo di un dispositivo temporal computing, cioè un microprocessore AP7000, in grado di eseguire una distribuzione

embedded di Linux. Dal lato FPGA sono presenti due moduli: il frame manager e il control manager. Il frame manager si occupa della realizzazione dei pacchetti da inviare all'AP7000 organizzando i frames in gruppi di elementi. Viene utilizzata una tecnica di buffer a ping-pong. Due memorie vengono utilizzate una in lettura e una in scrittura. Mentre una memoria viene riempita coi frame provenienti dal sistema di acquisizione, l'altra memoria viene collegata all'AP7000 affinche' possa essere letta attraverso un bus di comunicazione parallelo chiamato EBI. Ciascuna memoria possiede un control byte che serve a riconoscere lo stato in cui essa si trova (nuova, già letta, bloccata perchè in scrittura). La natura parallela del bus EBI permette trasferimenti ad alta velocità (nell'ordine di 30 MB/s). A livello del dispositivo AP7000 ho sviluppato un device driver per gestire il bus EBI e due applicazioni, una per l'invio dei frames e l'altra per la gestione dei parametri di configurazione del register file. Come in ogni moderno sistema operativo la memoria utente è virtualizzata e a livello utente non è possibile accedere allo spazio di indirizzamento fisico. E' stato quindi necessario sviluppare un device driver in grado di permettere alle applicazioni di interfacciarsi con le periferiche fisiche. Il driver da me sviluppato si occupa di fare da ponte tra il dispositivo FPGA e l'applicazione che si occupa della comunicazione via ethernet con il PC.
In particolare il driver, a fronte di una richiesta GET FRAMES controlla il control byte sulla FPGA. Appena esso è posto ad uno significa che sono disponibili almeno 100 frames da scaricare. Il driver legge i frames accedendo al base address a cui è mappata l'FPGA e ne copia il contenuto nella memoria USER. L'applicazione per la trasmissione dei frames via rete crea un server sulla porta 8008 TCP e attende una connessione da un client. Appena si istaura una connessione il server entra in modalità streaming e continua a richiedere frames al server ed ad inviarli (se disponibili) al client. In modo simile l'applicazione di controllo inizializza un server telnet-style al fine di attendere i comandi di configurazione dal PC.

Capitolo 1

La matrice di rivelatori

L'obiettivo del progetto di ricerca a cui ho collaborato è lo sviluppo e la realizzazione di uno strumento scientifico per spettroscopia gamma per applicazioni spaziali. In una prima fase del progetto, precedente a questo lavoro di tesi, sono state definite le caratteristiche del sistema di rivelazione. Non è obiettivo di questo lavoro l'analisi accurata delle soluzioni adottate; riporterò quindi nei successivi paragrafi solo un accenno alle problematiche affrontate nella fase di progetto.

Energy range	150keV – 15MeV
Active area of the SDD photodetection plane	78mm x 81mm
Dead area of the SDD photodetector	≤ 20%
QE of the SDDs	80% @ 380nm
ENC single SDD @-20°C (including ball.def.)	≤ 25e- rms
Expected resolution of gamma detector @662keV	≤ 3%
Operating temperature	-20°C
Cooling	Peltier
# readout ASICs	3
# channels per ASICs	27
total # channels (used for useful signals processing)	69
count rate capability	≤ 10000 counts/s

Tabella 1-1 Specifiche rivelatore

In tabella 1.1 vengono riportate le specifiche dell'unità di rivelazione:

L' obiettivo primario dello strumento è dunque la rivelazione di eventi gamma a basso rate in un range energetico decisamente esteso.

Una delle prime scelte effettuate in fase di progettazione è stata la scelta del cristallo scintillatore che è ricaduta sul Bromuro di Lantanio (LaBr3) che primeggia su tutti gli altri scintillatori per la luminosità e la risoluzione temporale.
Emettendo un numero elevato di fotoni per keV di radiazione incidente, garantisce un'eccellente risoluzione energetica; l'elevata efficienza è determinata dalla sua alta densità. Nella prima parte del progetto ci si è concentrati inoltre sullo studio del tipo di SDD da utilizzare.
Da questi studi preliminari incentrati sulle performance spettroscopiche e dalle specifiche di assemblaggio meccanico del sistema, è emerso che la soluzione migliore è una sottomatrice quadrata di area attiva 24x24mm² con 1mm di regione morta ai bordi, portando quindi la dimensione totale a 26x26mm². In

particolare nel caso di utilizzo di uno scintillatore cilindrico di dimensioni 3"x3" sono necessarie nove di queste matrici.
Nelle ipotesi di montaggio qui considerate , abbiamo preso in considerazione 1mm aggiuntivo su ogni lato per consentire il bonding elettrico dell'elettrodo di back. Per mezzo di attente simulazioni è emerso che la regione morta complessiva è nell'ordine del 15% in accordo con le specifiche di progetto.
Il passo successivo è stata la scelta del numero di detector in cui dividere ogni sottomatrice. Tale scelta è avvenuta in funzione del rumore elettronico del singolo SDD.
Una prima soluzione consisteva nel dividere l'area totale in 4 SDD aventi ciascuno un'area complessiva di 12x12mm^2. Questa ipotesi, che garantiva vantaggi per quanto riguarda la complessità dell'elettronica di lettura a causa della dimensione eccessiva del singolo SDD, era caratterizzata da tempi di drift di raccolta all'anodo molto grandi, elemento da non sottovalutare durante lo studio del rumore. Infatti accanto alla valutazione del rumore elettronico intrinseco, il peggioramento del rumore dovuto al deficit balistico associato al tempo di deriva del rivelatore gioca un ruolo molto importante. Si è quindi deciso di adottare una soluzione caratterizzata da 9 SDD di dimensioni minori.

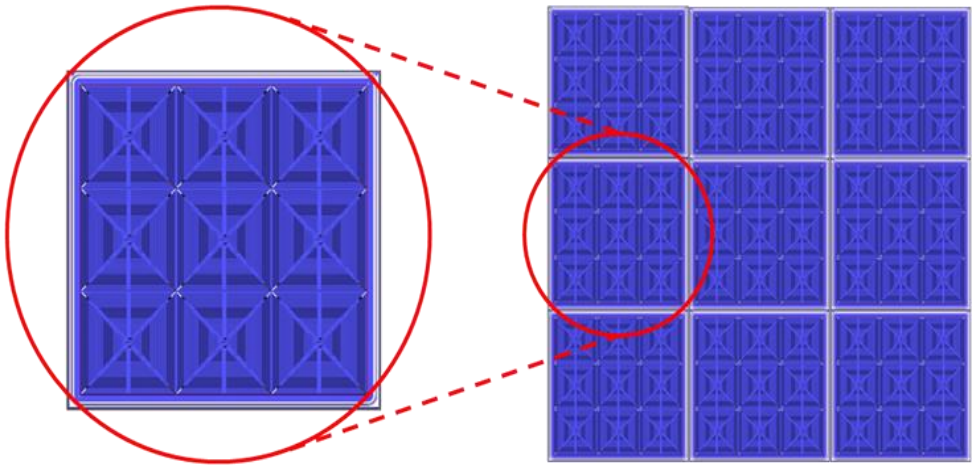


Figura 1-1 A destra la matrice di 81 SDDs, a sinistra un dettaglio della singola sottomatrice da 9 SDDs

La dimensione risultante del singolo SDD è quindi di 8mm. Il tempo di raccolta della carica, che dipende in prima approssimazione dal quadrato della lunghezza, è così molto più corto e di conseguenza anche il rumore elettronico è meno influenzato dal deficit balistico. Ovviamente lo svantaggio di questa soluzione è l'aumento del numero

di unità, 81 in totale. Di queste solo 69 risulteranno utili per il segnale mentre ad ogni angolo saranno presenti 3 detector che non riceveranno luce dallo scintillatore.

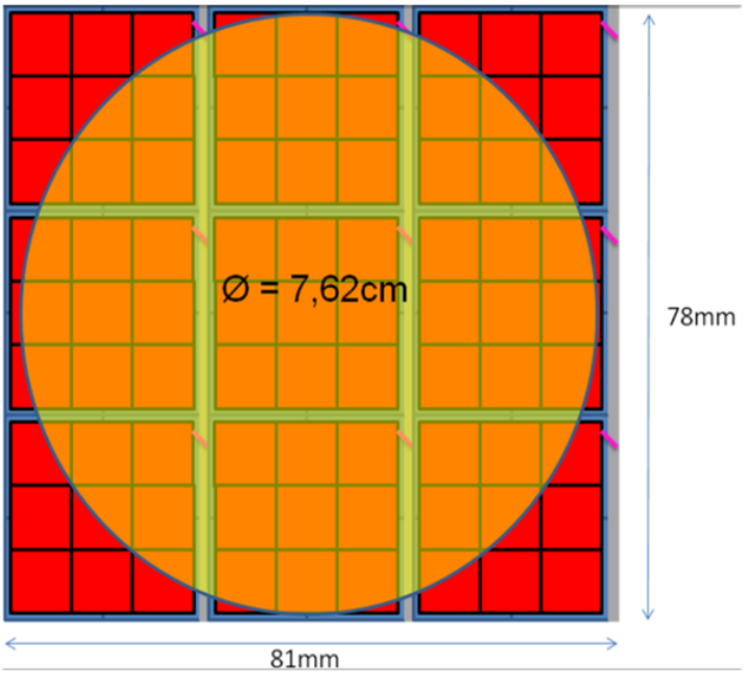


Figura 1-2 Immagine della matrice dei detector a cui è sovrapposto lo scintillatore; Notare come solo 69 SDD su 81 sono interessati da radiazione incidente

Studiamo nel dettaglio ora il singolo SDD: esso è costituito da un anodo n+ di forma di circolare posto al centro del detector, intorno al quale vengono realizzate impiantazioni p+, concentriche all'anodo, che rappresentano i catodi.
All'aumentare dalla distanza dal centro la forma dei catodi tende a squadrarsi.
Catodi contigui sono collegati per mezzo di una resistenza integrata che fa parte di un Voltage Divider che si occupa di polarizzare con tensioni via via crescenti i catodi da quello più esterno a quello più interno.

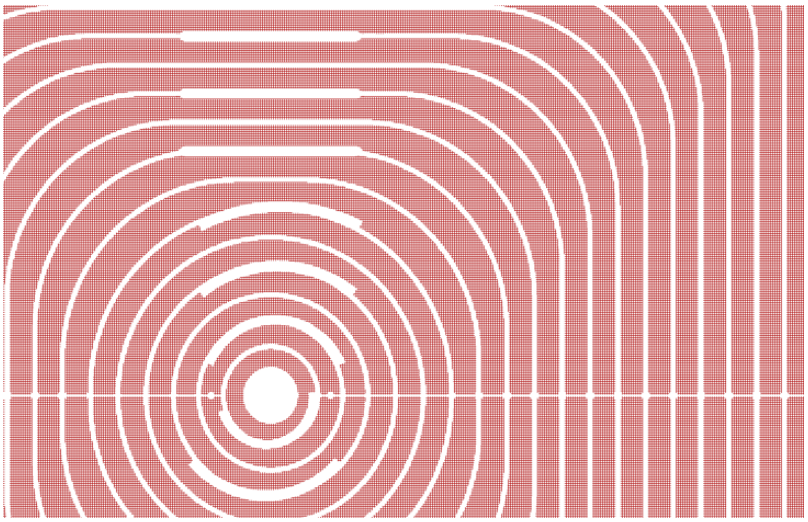


Figura 1-3 Dettaglio di un SDD in cui si nota la tendenza dei catodi a squadrarsi man mano che aumenta la distanza dal centro

Solo due tensioni vengono fornite dall'esterno: quella al primo e all'ultimo catodo; Quest'ultimo, essendo il più esterno, è in comune con tutti e 9 gli SDD.

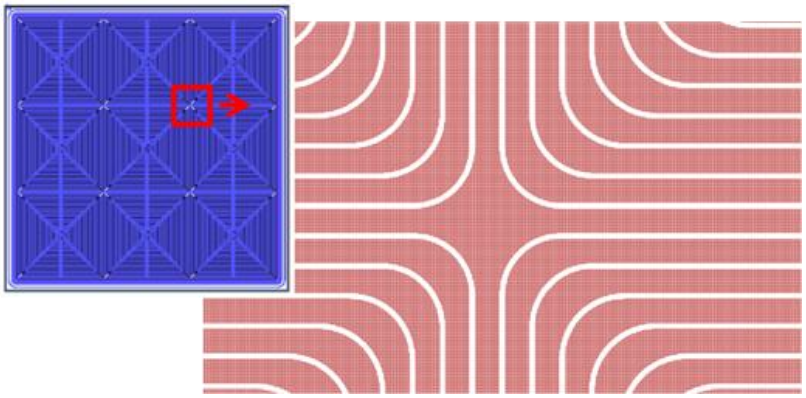


Figura 1-4 Dettaglio dell'ultimo catodo in comune per i 9 SDD della sottomatrice

La resistenza del voltage divider è stata impostata in modo tale da avere una corrente totale che scorre nel partitore nell'ordine dei 1-10μA. I fotoelettroni, nell'SDD, sono generati uniformemente sulla superficie del detector; pertanto la formazione del segnale dipende dal tempo di deriva della carica verso l'anodo. E' evidente che le cariche create nelle regioni più periferiche del detector impiegheranno un tempo maggiore e, in prima approssimazione, il tempo di deriva per un SDD di forma quadrata avente lato di lunghezza L vale:

$$t_{drift\ (max\ on\ L/2)} = \frac{(L/2)^2}{\mu V_{depletion}}$$

Questa formula risulta una rude approssimazione non solo perché il percorso lungo la diagonale ha lunghezza maggiore di L/2, ma bensì perché ipotizza che il campo elettrico sia costante lungo tutto il percorso. Attraverso delle simulazioni eseguite su un singolo detector di lato 1cm, abbiamo notato come la raccolta della carica all'anodo si può suddividere in 3 fasi ben distinte:
Una prima fase che si esaurisce velocemente (400ns) è caratterizzata dalla raccolta della carica nella zona centrale. Essa è responsabile della crescita lineare del segnale.
Nella fase successiva si ha la raccolta della carica della zona immediatamente successiva che è responsabile del decadimento non lineare del segnale (durata 1.5µs).
Nella terza fase la carica assorbita ai bordi del detector origina una decadimento esponenziale nella raccolta all'anodo che può considerarsi esaurito dopo 4µs e che dipende fortemente dalla dimensione del detector .

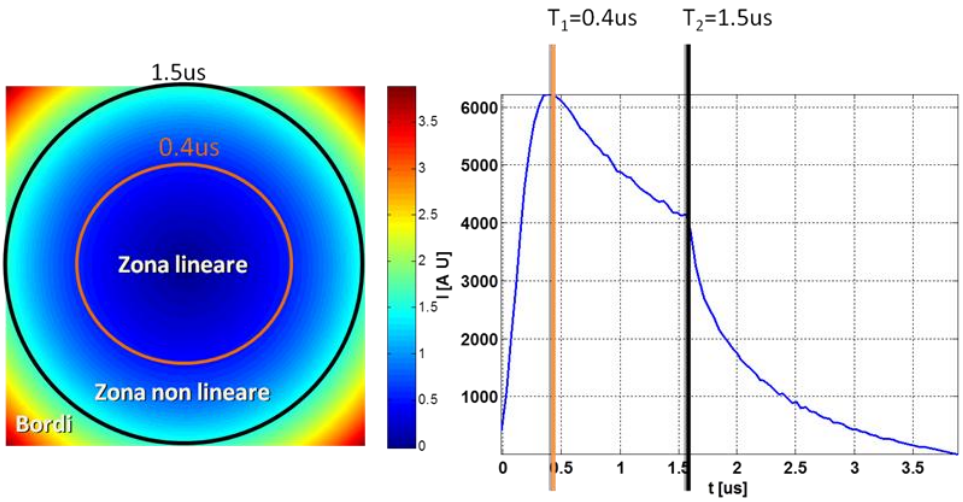


Figura 1-5 Zone di assorbimento del singolo SDD e forma del segnale d'uscita per un detector da 1cm2

Il sistema di rivelazione

L'unità base del modulo di rivelazione è costituita da un matrice monolitica di 9 SDD opportunamente montati su un supporto di ceramica che comprende anche il primo stadio dell'elettronica (Preamplificatore). Il collegamento del contatto dell'anodo di ogni SDD ad un preamplificatore CMOS, posizionato in prossimità del detector stesso, è possibile per mezzo di nove fori attraverso la ceramica. La

scheda di ceramica utilizzata è a due layer: su di un lato come detto sono posizionati i preamplificatori, l'altro è utilizzato per il routing delle alimentazioni.

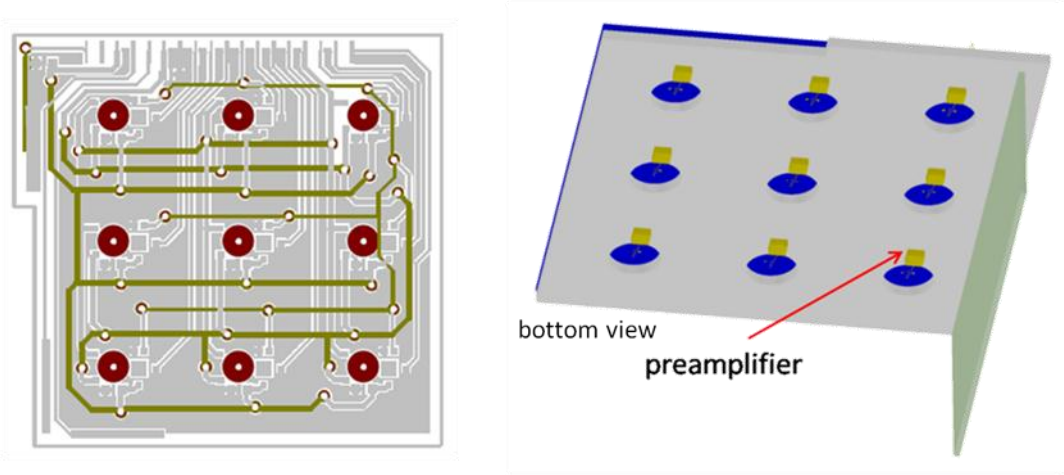


Figura 1-6 La figura mostra il layout della ceramica dove lo strato superiore è rappresentato in colore grigio mentre quello inferiore in verde. A destra si può notare come i preamplificatori siano posizionati sul bottom, a ridosso dei fori.

Per proteggere gli ASICs da eventuali danni e da accoppiamenti capacitivi viene fissata una copertura in rame sotto la ceramica.
Questa copertura è ricavata da un blocco di rame con tre cavità ciascuna che corre lungo una fila di tre amplificatori.

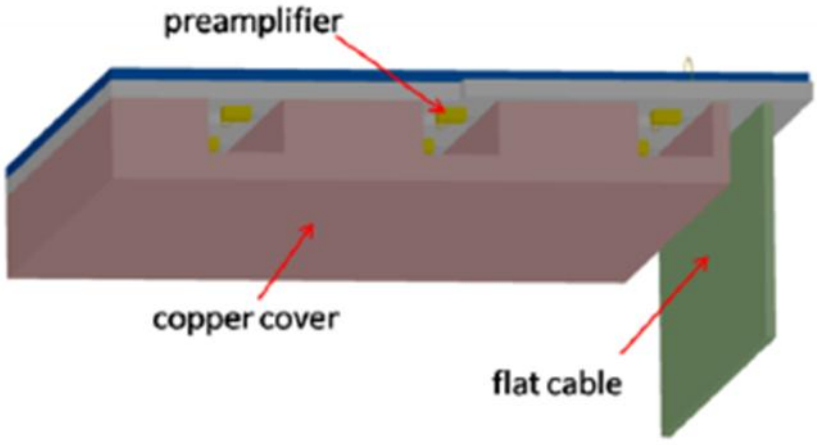


Figura 1-7 La figura mostra il blocco di rame protezione dei preamplificatori con le tre cavità

Queste camere si aprono da un lato per permettere la circolazione dell'azoto e quindi per evitare il surriscaldamento dei dispositivo. Particolare attenzione è stata posta allo studio del comportamento

termico del sistema: la ceramica utilizzata per la scheda rivelatori ha una conduttività termica di 180 W/mK e anche il foglio adesivo utilizzato per incollare tutte le parti è caratterizzato da buona conducibilità termica.
Studi preliminari sono stati fatti con un simulatore della Comsol Software per valutare la quantità totale di calore da estrarre e per poter scegliere il sistema di raffreddamento a celle di Peltier che meglio si adattava alle nostre esigenze.
L'intero sistema composto da scintillatore e piano di rivelazione viene portato alla temperatura di funzionamento di -25 °C, al fine di ridurre il leakage. La quantità totale di energia che deve essere estratta dal sistema di raffreddamento Peltier è di circa 4W.
Considerando che ogni Peltier deve estrarre una potenza corrispondente a 4W/9=450mW, dove 9 è il numero di moduli, il delta di temperatura di 50°C (necessaria per passare da 25 ° C a -25°C) potrebbe essere raggiunto polarizzando il peltier alle condizioni minime di funzionamento che permettono infatti di estrarre 0.5W e raggiungere un delta T di 50 °C con il minimo di corrente e voltaggio richiesto (1A, 3.6V).
Questo lascia un margine di sicurezza per qualche aggiustamento nelle reali condizioni operative. Il dissipatore di calore in rame è stato invece progettato per estrarre una potenza nell'ordine dei 50W, che è la somma della potenza da estrarre dalle celle Peltier (4W) più lpa potenza generata per la polarizzazione delle celle Peltier stesse (3,6 W x 9 = 32 W).

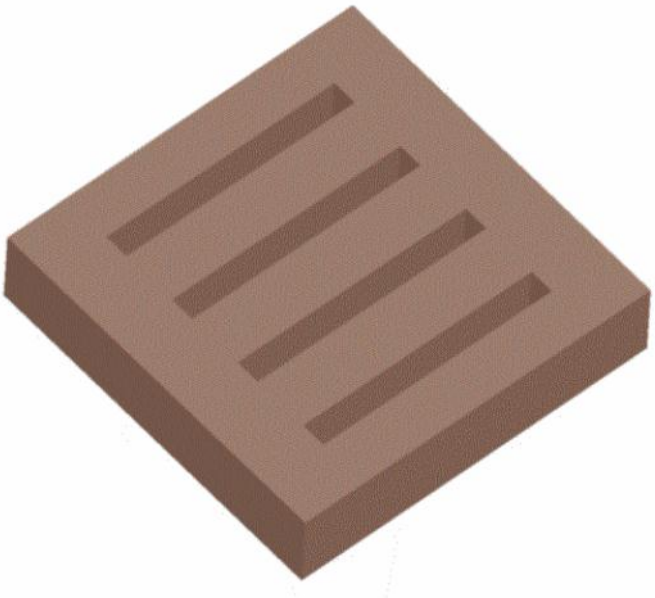


Figura 1-8 La figura mostra il dissipatore in rame che presenta 4 fessure che permettono il passaggio dei segnali dalla scheda di ceramica ad un'altra posta immediatamente sotto il dissipatore stesso.

Per consentire il montaggio di ogni unità sopra il dissipatore in rame, deve essere aggiunto un altro supporto, anch'esso in rame , incollato sul lato caldo della cella di Peltier.
Le dimensioni di questo supporto sono le stesse del modulo SDD con un lato di apertura per consentire il passaggio ai cavi di segnale e alimentazione del peltier. In particolare un sistema di perni blocca ogni unità da quella successiva. Questo produce un allineamento delle unità stesse e ci aiuta durante il montaggio dove tutte le unità devono essere fissate al dissipatore di calore.
Sotto il dissipatore in rame è presente una prima scheda elettronica, collegata mediante cavi flat alle schede di ceramica, che fornisce le tensioni d'alimentazione dei detector e riceve i segnali in uscita dal preamplificatore. Proprio quest'ultimi sono reindirizzati verso un ulteriore scheda su cui è presente l'elettronica di front end.
Infine fra scintillatore e piano di rivelazione è stato inserito uno strato di spacer ottico al fine di ridurre il rischio di danneggiamento dei bonding su scheda; lo spessore di questo strato è nell'ordine dei 200μm. La scelta del grasso ottico, utile allo scopo è ricaduta sul BC-634A, che garantisce facilità di applicazione e allo stesso tempo alta resistenza ad eventuali lacerazioni in fase di manipolazione. L'intero sistema è stato inserito in un contenitore in alluminio in atmosfera controllata. Ciò viene fatto principalmente per evitare danni causati da fenomeni di condensa durante le procedure di raffreddamento. Per questo motivo il contenitore è saturo di azoto.

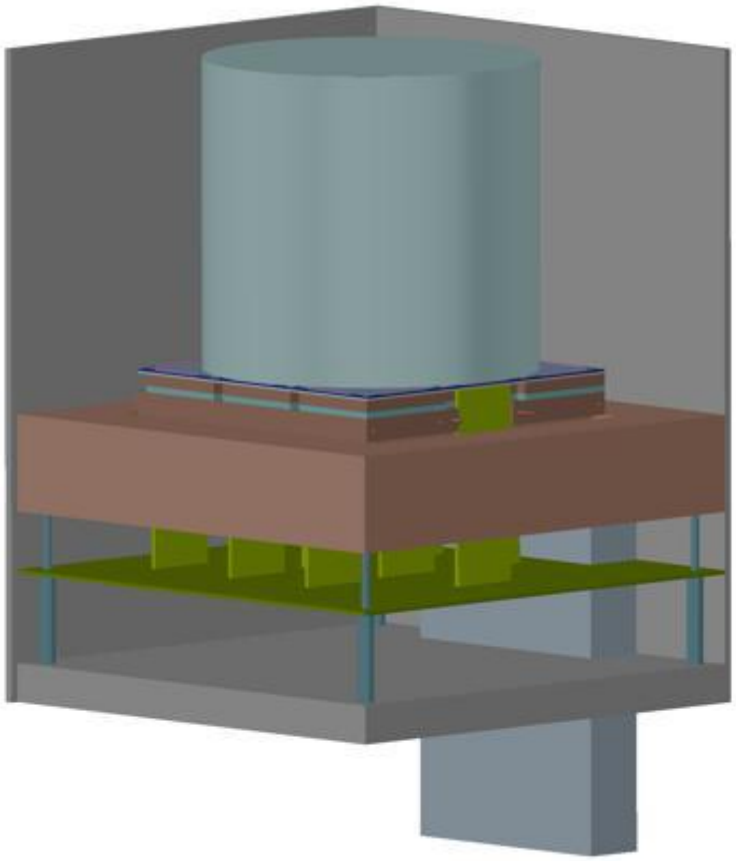


Figura 1-9 Immagine del box in alluminio che racchiude il sistema di rivelazione.

Il preamplificatore di carica

Il primo stadio del front-end analogico è costituito dal preamplificatore di carica che ha il compito di convertire la carica fotogenerata degli SDD in tensione.
I requisiti che deve avere un buon amplificatore di carica sono alto guadagno e basso rumore. Lo schema semplificato dell'amplificatore utilizzato in questo progetto è:

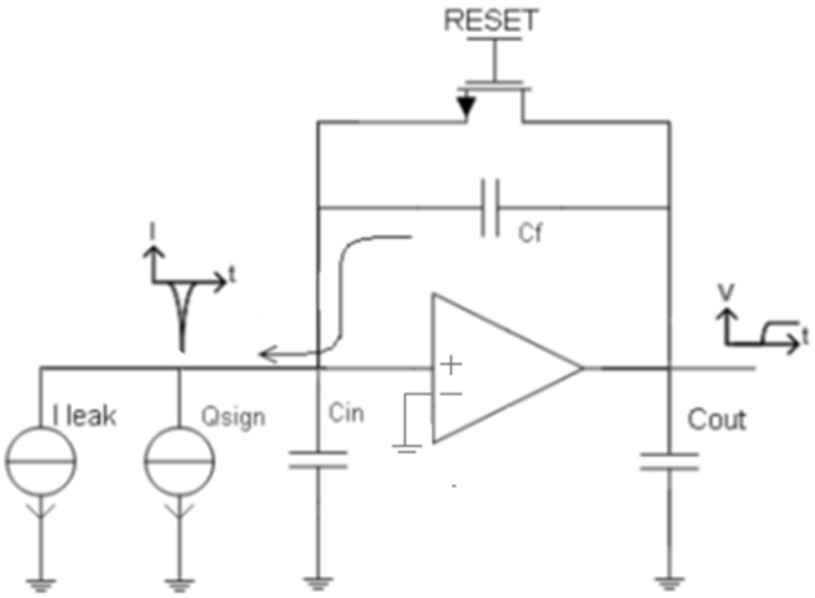


Figura 1-10 Schema semplificato dell'amplificatore utilizzato in questo progetto

Questo tipo di stadio converte il segnale di corrente in un segnale di tensione mediante l'integrazione del flusso di cariche nella capacità di feedback.
Ciò avviene perchè l'effetto della retroazione è quello di mantenere il morsetto negativo dell'operazionale un nodo di terra virtuale. Le cariche generate dal rivelatore vengono integrate nella capacità e il segnale in uscita si presenta come una sovrapposizione di gradini di ampiezza pari a:

$$V_{OUT} = \frac{Q}{C_f}$$

Anche la corrente di leakage del detector viene integrata nella capacità di feedback; gli effetti dell'integrazione di segnale e leakage sono mostrati in Figura 1.11.

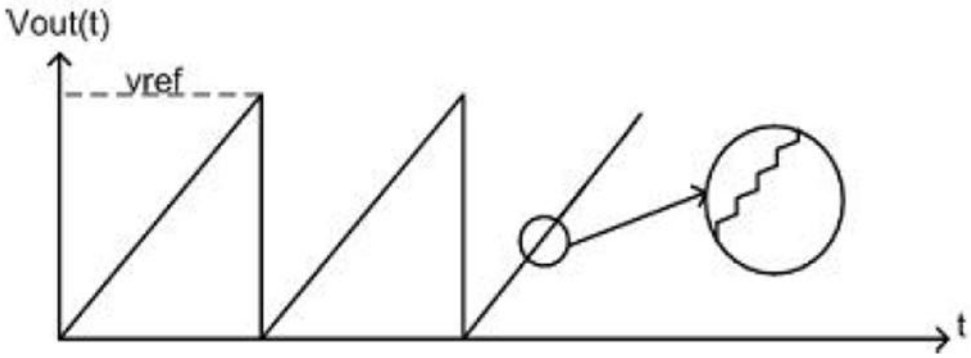


Figura 1-11 Andamento della tensione all'uscita del preamplificatore.

Considerando l'integrazione del leakage sulla capacità di feedback è possibile ricavare la pendenza della rampa che sarà uguale a:

$$\frac{dV}{dt} = \frac{I}{C_f}$$

Operativamente durante le sessioni sperimentali, il leakage viene misurato tramite oscilloscopio.
Valori tipici di questa corrente, alla temperatura di -20C, si attestano intorno ai 24pA. Ne risulta che la pendenza della rampa con una capacità di feedback stimata in 23fF è:

$$\frac{dV_{OUT}}{dt} = \frac{I_{Leakage}}{C_f} \sim \frac{24pA}{23fF} \sim 1\frac{V}{ms}$$

Il principale effetto dell'integrazione del leakage è quello di fare saturare l'amplificatore. E' quindi necessario scaricare periodicamente la capacità di feedback.
In questo progetto il preamplificatore è realizzato in tecnologia CMOS; il reset della C_f avviene con un interruttore N-Mos che cortocircuita la capacità.
Al fine di ottenere bassi livelli di rumore, questo amplificatore è stato progettato con uno stadio d'ingresso a PMOS che ha generalmente un rumore 1/f inferiore rispetto all'NMOS. Inoltre la dimensione del transistor, la sua capacità di gate e la sua corrente di polarizzazione sono state ottimizzate al fine di raggiungere le migliori prestazioni di rumore in presenza della connessione all'SDD.

Il range dinamico

Prima di iniziare la progettazione analogica della catena di rivelazione si sono resi necessari studi preliminari sulla dinamica del segnale in ingresso. L'obiettivo del progetto ESA è quello di ottenere una corretta rivelazione su un range energetico molto ampio, 150keV - 15MeV; sono state eseguite quindi delle simulazioni Montecarlo che hanno preso in considerazione il sistema composto da scintillatore e piano di rivelazione, la sua efficienza di conversione e la riflettività delle varie superfici su tutto il range energetico. I risultati di maggiore interesse sono stati ottenuti in corrispondenza dell'evento a energia minima (150Kev) e dell'evento ad energia massima (15 Mev).
Nel caso di illuminazione omogena del cristallo con eventi ad energia minima (150 Kev), si osserva chiaramente un cut-off intorno ai 100 elettroni che rappresentano quindi il segnale minimo che dovrebbe essere rilevato da un SDD. A questo punto è importante sottolineare che non appena un segnale da almeno un SDD supera questa soglia, tutti i segnali proveniente da tutti gli altri SDD (indipendentemente dal fatto che abbiano raccolto anche loro almeno 100 elettroni) entreranno in fase di lettura. Tale lettura dovrà tenere conto anche del livello di rumore del fotorivelatore (25-30e-rms) che quindi impedisce di rivelare segnali inferiori a questo limite. Al contrario si osserva che in presenza di un evento di energia massima (15 Mev) il valore massimo di elettroni raccolti è circa 30000.

Ne consegue, ragionando all'uscita del preamplificatore, che in presenza di un evento a massima energia, il ΔV di segnale sovrapposto alla rampa dovuta al leakage sarà pari a:

$$\Delta V_{OUT,PREAMP,MAX} = \frac{Q_{signal,MAX}}{C_f} \approx \frac{30.000 \cdot 1.6 \cdot 10^{-19}}{23fF} \approx 200mV$$

e a tale ampiezza si dovrà associare la massica dinamica all'uscita della catena analogica (come si vedrà nel capitolo successivo).
Per l'evento a minima energia si avrà invece:

$$\Delta V_{OUT,PREAMP,MIN} = \frac{Q_{signal,MIN}}{C_f} \approx \frac{100 \cdot 1.6 \cdot 10^{-19}}{23fF} \approx 0.7mV$$

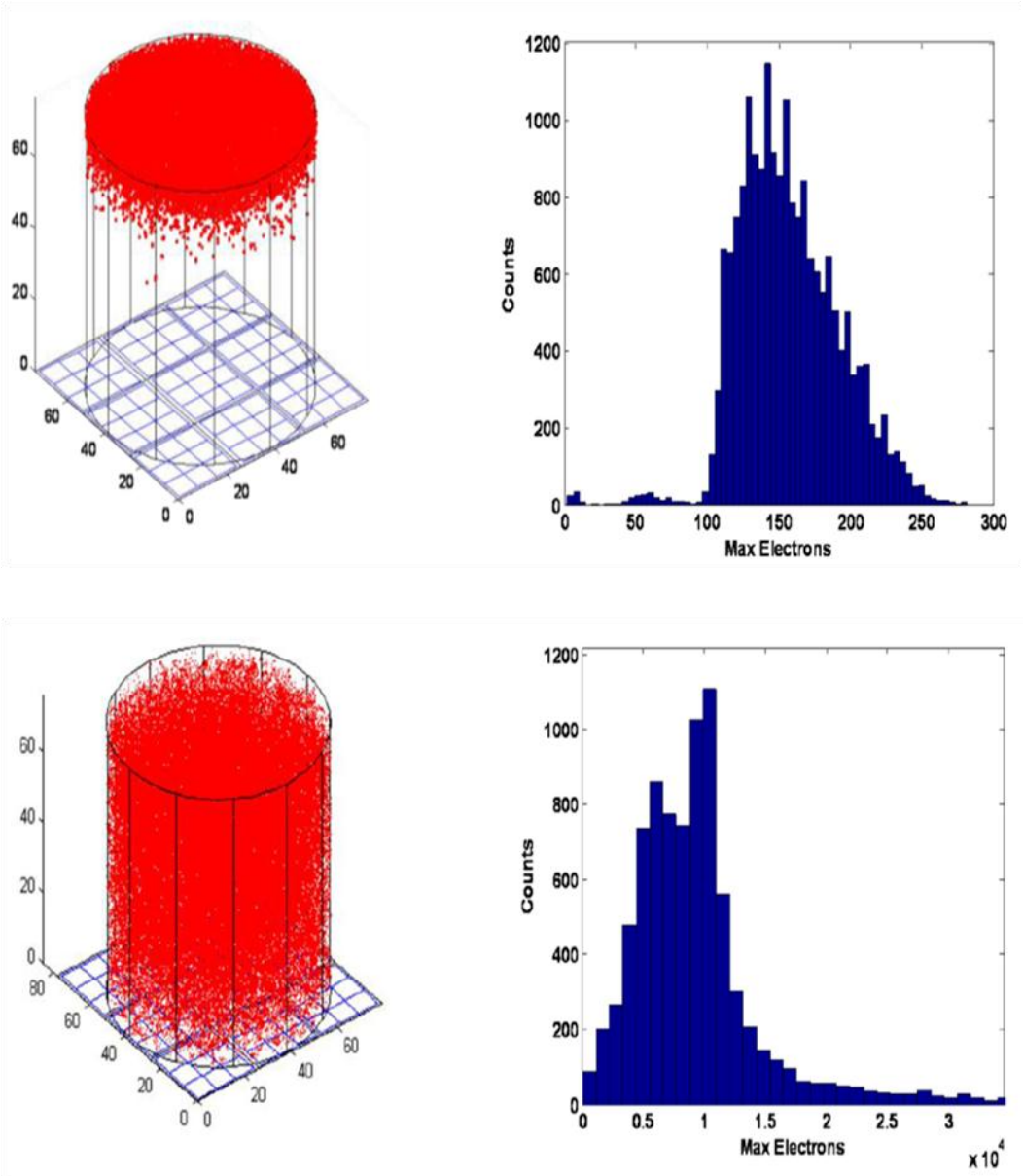


Figura 1-12 I grafici di sinistra mostrano i punti di interazione all'interno del cristallo mentre quello di destra la distribuzione statistica del numero massimo di elettroni raccolti in corrispondenza di eventi ad energia minima 150 Kev (in alto) e ad energia massima.

Capitolo 2

L'elettronica di read out

In questo capitolo è descritta l'elettronica di read-out realizzata per il progetto. Verranno dunque illustrati i principali blocchi che costituiscono il singolo canale analogico con attenzione marginale ai dettagli tecnici di progettazione.

Caratteristiche dell'elettronica di lettura

Nel progetto ESA il compito del restante front-end analogico è, dopo l'oppurtuna amplificazione da parte del preamplificatore di carica, di filtrare il segnale con un circuito formatore analogico (shaper) al fine di migliorare l'SNR e di campionare l'ampiezza per inviarla successivamente al sistema di acquisizione digitale (DAQ). Inoltre la catena analogica implementa un trigger per il DAQ e provvede al reset del preamplificatore.
Considerato il numero di canali da leggere (81), è stato scelto di suddividere la lettura dei rivelatori con tre ASIC da 27 canali ciascuno. Questa scelta è sembrata un buon compromesso tra costi e prestazioni. Un numero superiore di canali per chip avrebbe rappresentato infatti un rischio in termini sia di complessità del progetto sia in termini di probabilità di fallimento di un canale. Al contrario un numero troppo piccolo di canali per chip avrebbe comportato un eccessivo consumo di spazio ed una prevedibile difficoltà nelle procedure di test.
Come visualizzato il sistema può essere suddiviso in più elementi:

- canale analogico: costituito da uno Shaper, da un Baseline Holder e da un Peak Stretcher oltre che dalla logica associata di cui si discuterà in seguito;
- multiplexer: ha il compito di portare in uscita i 27 canali, uno alla volta;
- buffer d'uscita: posto a valle del MUX, fornisce la corrente necessaria per il pilotaggio dei parassitismi su scheda;
- Serial Peripheral Interface (SPI): interfaccia LVDS per la programmazione delle varie funzioni logiche dell'ASIC;
- riferimenti di tensione e corrente;
- logica di Trigger, per riconoscere l'arrivo dell'evento rivelarne il picco associato e fornire informazioni sul timing.

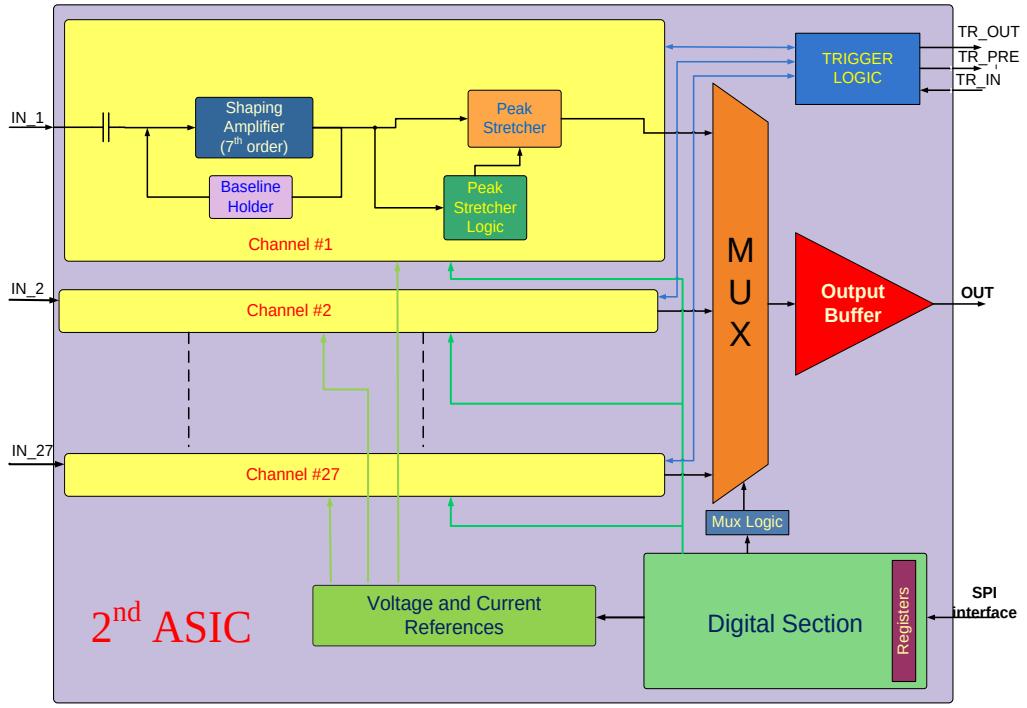


Figura 2-1: In figura viene rappresentato lo schema a blocchi dell'elettronica.

Il compito del Chip consiste quindi nell'individuare l'arrivo di un evento, formare gli impulsi di carica associati ai 27 SDD con un filtro opportuno, mantenere i valori di picco in ogni canale e fornire questi valori all'uscita verso l'acquisizione attraverso un MUX 27 ad 1 e un buffer d'uscita.
A contorno del funzionamento base appena descritto è presente una logica che ha il compito da un lato di rendere possibile la comunicazione con la scheda DAQ, e dall'altro di andare a modificare parametri importanti nella catena analogica quali il guadagno, il tempo di formatura, correnti e tensioni di riferimento per i diversi stadi.
Riporto in figura 2.2 un immagine del layout dell'ASIC completo. Il Chip è stato realizzato in tecnologia AMS (Austria micro system) a 0,35 μm.

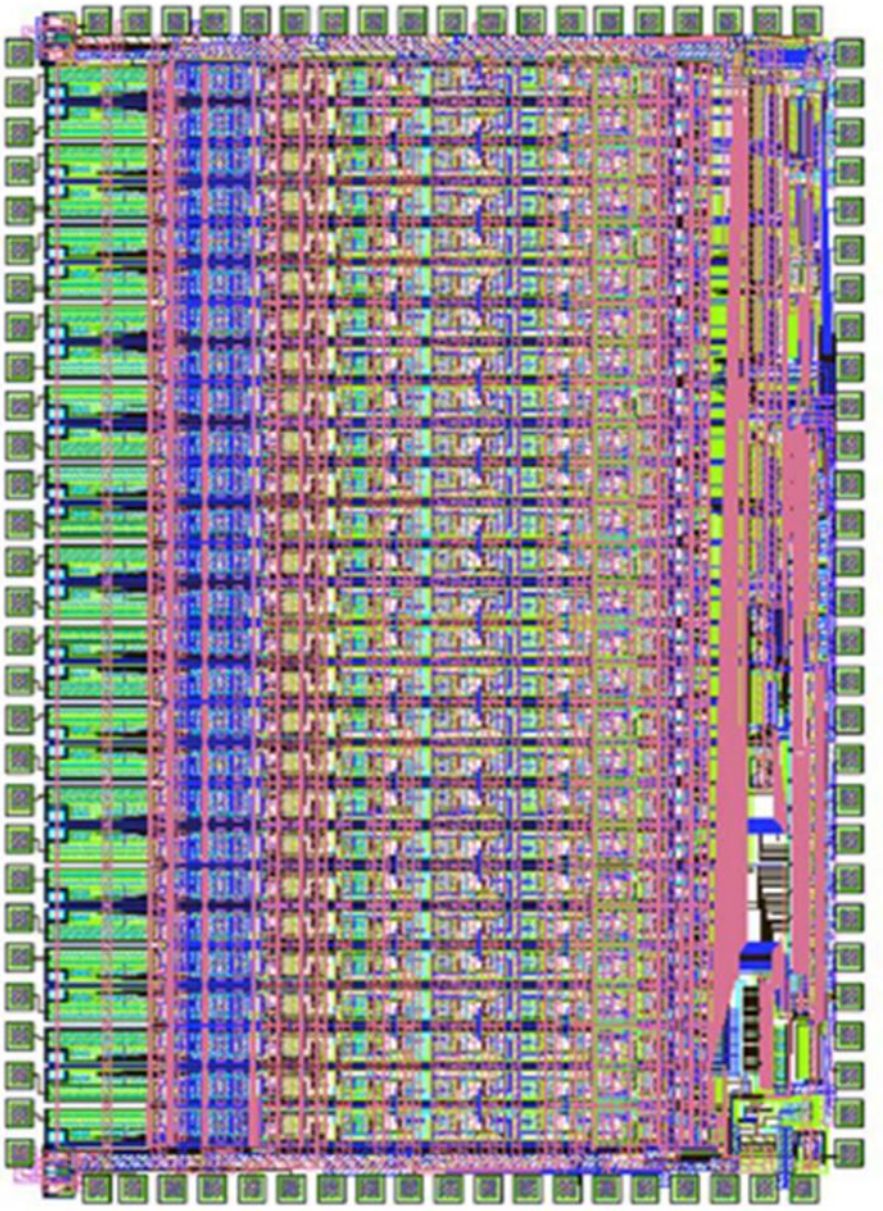


Figura 2-2. Layout dell'ASIC completo

Lo Shaper

Lo Shaper a poli complessi e coniugati è tipicamente realizzato dalla cascata di celle del 2° ordine.

Bisogna tuttavia ricordare che il preamplificatore, per come è realizzato, introduce un polo nell'origine o a bassissime frequenze nel trasferimento, e questa singolarità andrà compensata con un successivo stadio a derivatore approssimato che introduce a sua volta un polo reale a frequenza progettabile; ne consegue che avendo già a disposizione un polo reale nella catena sarà preferibile realizzare un filtro formatore di ordine dispari aggiungendo tante celle del 2° ordine

quante sono necessarie. Questa ragione spinge a scartare a priori i filtri del 4° e del 6° ordine dalla scelta.

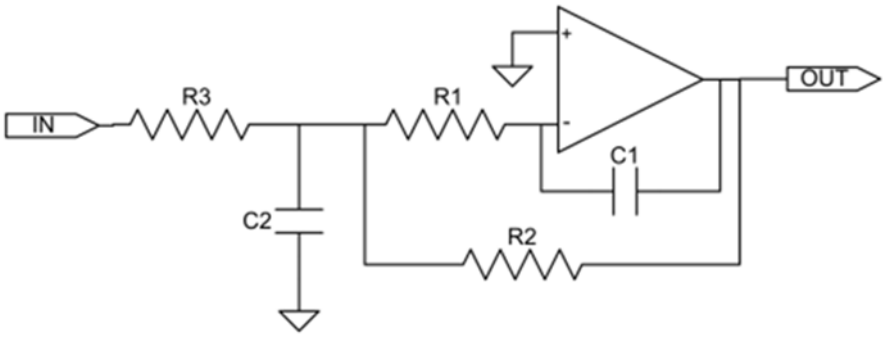


Figura 2-3 Schema circuitale della cella Multiple Feedback (MFB)

La scelta è infine ricaduta su uno shaper semigaussino del 7° ordine realizzato mediante 3 celle MFB (Multiple Feedback) a valle del derivatore in grado di fornire sufficienti gradi di libertà per ridurre le sorgenti di rumore circuitali e allo stesso tempo garantire una dinamica sufficiente per l'applicazione.

Il Baseline Holder

Il baseline holder è un circuito che svolge la funzione di stabilizzare ad una prescelta tensione di riferimento la linea di base del segnale all'uscita dello shaper; disporre di un tale circuito risulta assolutamente necessario dal momento che in sua assenza la linea di base subirebbe fluttuazioni significative a causa di errori di processo, missmatch e drift termici.

Il circuito deve fissare la tensione in DC all'uscita dello shaper senza però modificarne la fdt e quindi la caratteristica risposta semigaussiana all'impulso di corrente.

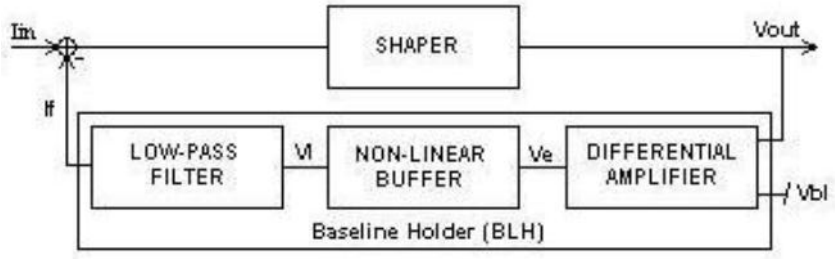


Figura 2-4 Schema a blocchi del baseline holder

Come visualizzato il sistema è composto da tre blocchi: un amplificatore differenziale, un buffer non lineare e un filtro passa basso.

Osservando l'effetto della retroazione negativa possiamo qualitativamente distinguere le seguenti due condizioni di

funzionamento: in DC la tensione Vout della linea di base viene confrontata con una tensione di riferimento Vbl e viene quindi prodotto un segnale errore Ve che sostiene una opportuna corrente If iniettata nello shaper affinché la differenza Vout − Vbl tenda idealmente a 0V; in AC il trasferimento If/Vl diviene sostanzialmente nullo inibendo l'azione del baseline holder e lasciando quindi inalterata la risposta dello shaper ad un impulso di corrente.
Alla luce di quanto detto, le funzioni di trasferimento di shaper e baseline holder devono essere della forma mostrata in figura 2.10.
Per comprendere la struttura del sistema evidenziamo innanzitutto i requisiti che devono essere soddisfatti dal sistema:

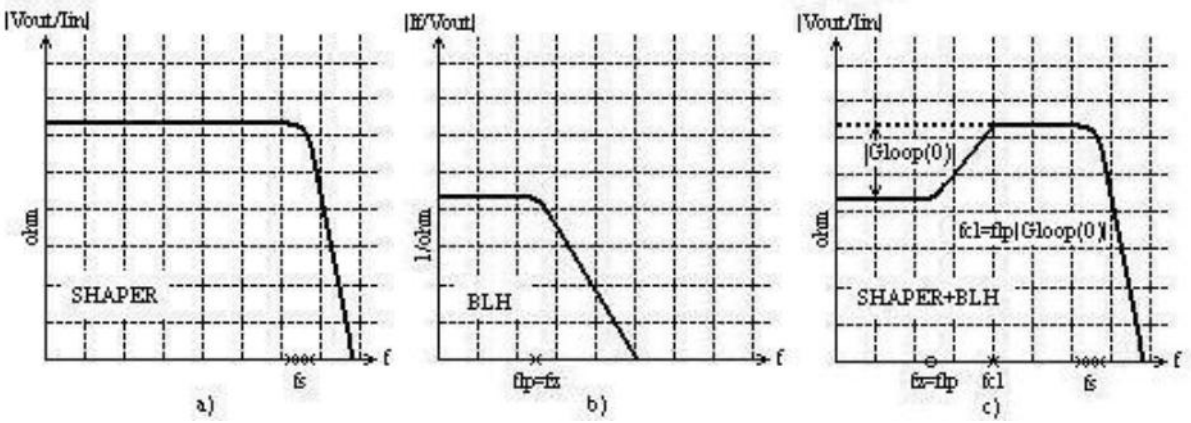


Figura 2-5 Funzioni di trasferimento: a) del solo shaper b) e c) del base line holder.

Osservando la terza figura è evidente che se il guadagno d'anello è elevato l'ingresso Iin e l'uscita Vout risultano sostanzialmente accoppiati in AC e il valore in DC di Vout deve essere all'incirca nullo con l'inconveniente che la linea di base subisce uno shift Δ(Vbl) negativo (fig. 2.6) crescente con l'aumentare del rate di eventi.

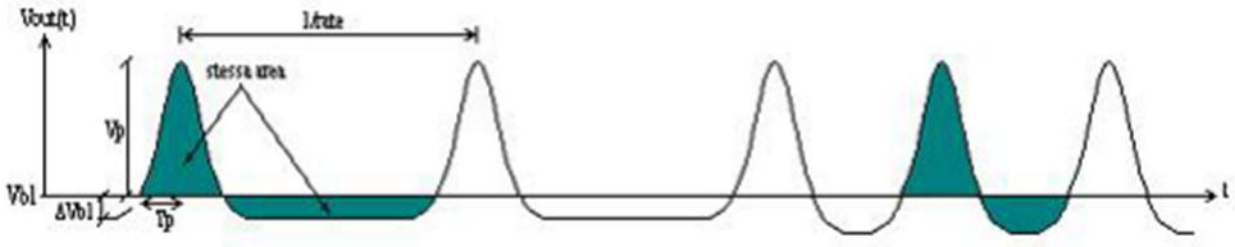


Figura 2-6 Shift della linea di base a causa dell'accoppiamento AC.

I requisiti che devono essere soddisfatti sono i seguenti:

- Piccolo errore Vout – Vbl;
- Stabilità e trasferimento ingresso-uscita il più possibile inalterato ;
- Compensazione dello shift legato all'accoppiamento AC;

Il primo requisito rimanda ad un elevato valore del Gloop(0), si desidera avere uno shift a regime di pochi mV rispetto alla tensione di riferimento.
Il secondo requisito impone un limite sulla frequenza massima a cui collocare il polo ad anello chiuso fcl e più precisamente più di una decade prima dei poli dello shaper; infatti ad una sola decade i poli (nel nostro caso 7) dello shaper contribuiscono significativamente a ridurre il margine di fase dell'anello.
Senza la presenza di un buffer non lineare, l'accoppiamento AC impone che il valore medio della forma d'onda Vout(t) sia nullo ed ovvero che la linea di base subisca uno shift negativo.
L'espressione di tale shift è la seguente:

$$\Delta(\mathrm{Vbl}) = V_P T_P \mathrm{rate}$$

Essa è facilmente ricavabile approssimando a triangolo la risposta semigaussiana dello shaper ed imponendo l'uguaglianza fra l'area sovrastante e l'area sottostante la tensione di riferimento Vbl.
Al fine di ridurre l'entità di tale shif si introducono due linearità:
Dimensionando il guadagno dell'amplificatore differenziale in modo che esso saturi all'arrivo di una gaussiana;
Introdurre un buffer non lineare in cascata all'amplificatore differenziale.

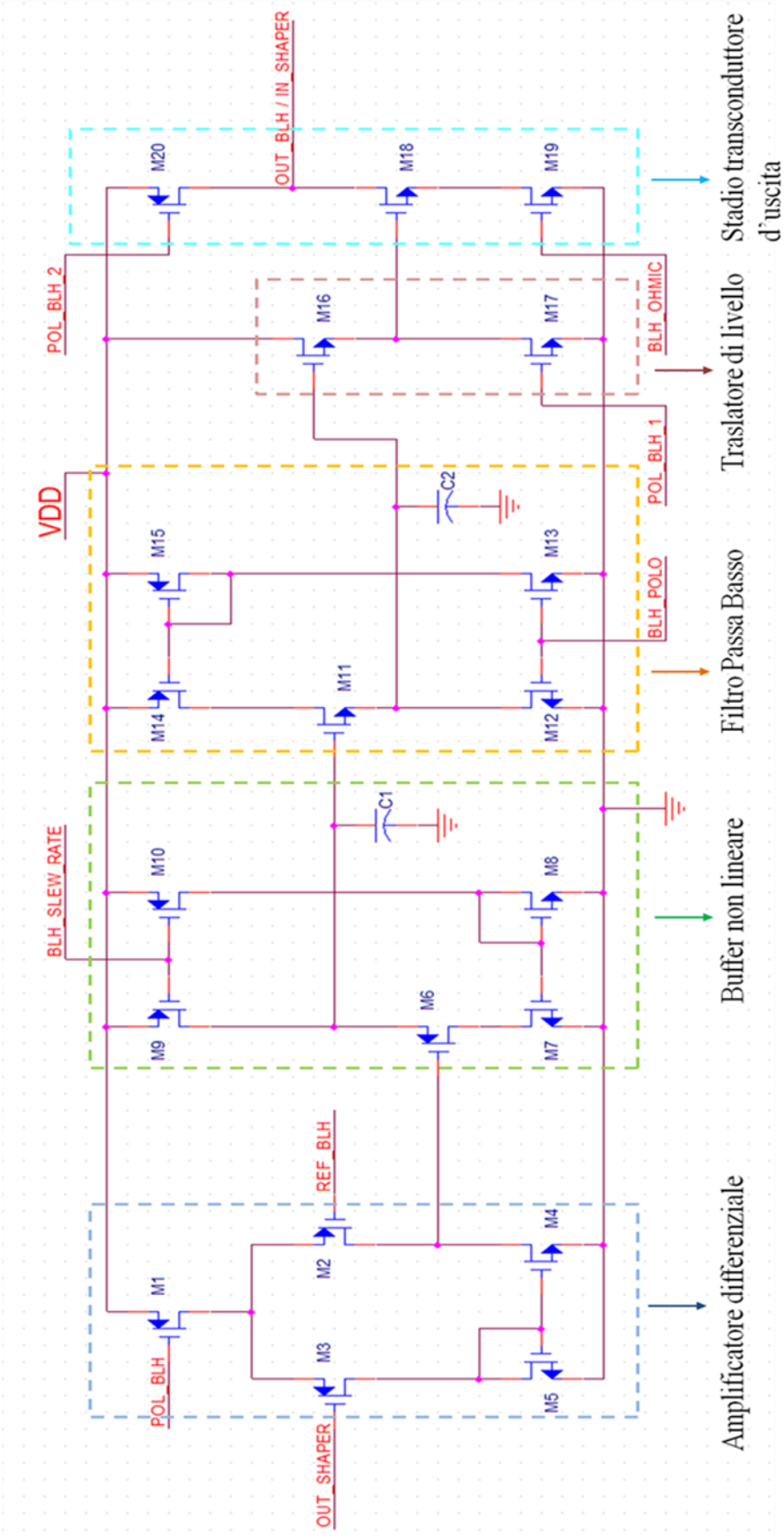


Figura 2-7 Schematico del baseline holder progettato per il chip ESA.

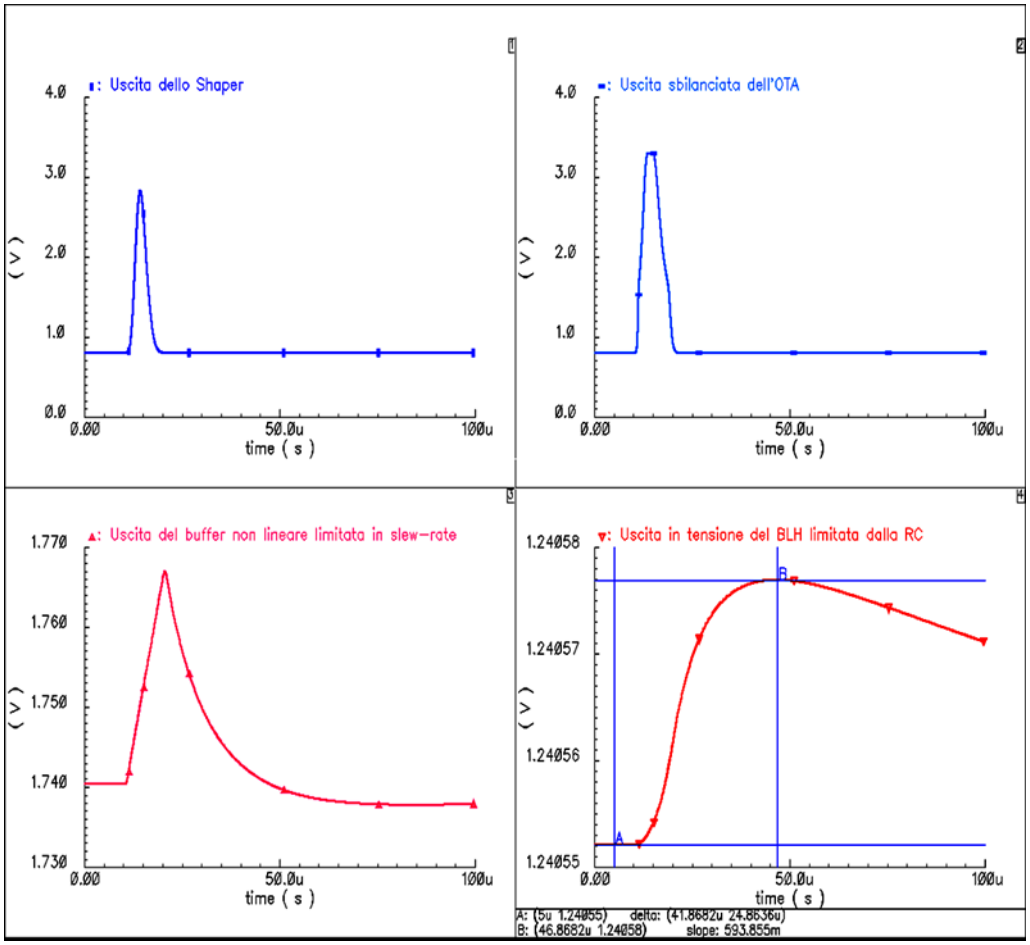


Figura 2-8 Segnali nei nodi intermedi del BLH in corrispondenza del segnale semigaussiano all'uscita dello Shaper.

Il BLH, per come è stato progettato, ci permette di aumentare la dinamica disponibile al nodo d'uscita dello Shaper: questa idea, che ha caratterizzato e distinto il Chip ESA dai precedenti realizzati, merita di essere approfondita.
Come visto precedentemente, il dynamic range previsto per il progetto ESA si estende da energie di 150KeV a 15MeV a cui si associano rispettivamente ampiezze di picco massime all'uscita degli Shaper pari a 7mV e 2V.
Supposto che le singole celle dello Shaper abbiano un riferimento ai nodi non invertenti degli OTA di 1.65V, verrebbe intuitivo fissare questo riferimento anche per il Baseline Holder (VBL) vincolando la tensione in DC dell'uscita a tale valore; in questo modo però si andrebbe a perdere metà della dinamica potendo sfruttare solamente la metà superiore da 1.65V a 3.3V.
Si è deciso quindi di mantenere i riferimenti degli Shaper a 1.65V, limitandosi a variare il riferimento del BLH portandolo a valori più bassi (a partire da 400mV, comunque regolabili).

La possibilità di portare ad un minimo di 350/400mV il riferimento del Baseline Holder e rivelare ampiezze di picco fino a 3.1V rende la dinamica all'uscita dello Shaper quasi rail to rail e potrebbe garantire l'acquisizione di segnali energetici superiori alla soglia massima di 15MeV.

Reset, Inhibit e Kill del canale

Nei paragrafi precedenti si è spiegato come il leakage del detector venga integrato sulla capacità di feedback del preamplificatore generando in uscita un segnale a rampa: il meccanismo di Reset impedisce a questa tensione di superare valori oltre i quali gli stadi successivi rischierebbero di lavorare in maniera errata. La logica che permette di rilevare il superamento di una determinata soglia da parte dell'uscita del preamplificatore è inserita all'interno dell'ASIC e interessa ovviamente il nodo d'ingresso del singolo canale analogico: si tratta in sostanza di un comparatore a soglia programmabile la cui uscita è un segnale di trigger, chiamato per ragioni logiche TR_PRE; Questo segnale, generato da ogni singolo canale dei 27 che costituiscono l'ASIC, viene inviato esternamente al Chip e quindi all'acquisizione (dopo averne fatto la OR dei 27 segnali internamente); l'FPGA risponde generando un segnale di RESET_PRE che attraverso il bus e il chip ASIC raggiunge i preamplificatori sotto la matrice dei detector, resettandone la carica accumulata sulla capacità di feedback.
L'uscita del preamplificatore subisce una brusca transizione verso il basso: in assenza di accorgimenti, il fronte di discesa si presenterebbe invariato a valle della capacità di disaccoppiamento tra preamplificatore e shaper, nodo in cui afferiscono sia l'uscita del BLH che l'ingresso invertente dell'OTA del 1° stadio dello Shaper.
Una dopo l'altra, le celle dello Shaper vedrebbero schiantarsi a massa la propria uscita a seguito del propagarsi del fronte nella catena: ben presto lo sbilanciamento interesserebbe il Baseline Holder e i suoi nodi intermedi, generando l'apertura dell'anello e la conseguente risalita delle tensioni dello Shaper verso il valore di 1.65V; ben più grave sarebbe il seguito: quando la breve fase di reset termina, il BLH è completamente sbilanciato e la costante di tempo di carica/scarica associata al filtro passa basso presente a valle del buffer non lineare, non permette di recuperare la condizione di corretto funzionamento in tempi brevi; si rischia in altri termini di avere un tempo morto a seguito di ogni operazione di reset di decine o centinaia di μs, ovviamente inaccettabile per l'applicazione.

Si spiega quindi l'esigenza di inibire sia lo Shaper che il Baseline Holder ogni qualvolta si vadano a resettare i preamplificatori.
Questo accorgimento consente di recuperare velocemente, in circa 10µs, la condizione di corretta polarizzazione dello Shaper. Tuttavia, se nessun provvedimento venisse preso per il BLH, una variazione del nodo d'uscita e del nodo d'ingresso dello Shaper sarebbe più che sufficiente per sbilanciare l'amplificatore differenziale e lo stadio d'uscita del Baseline, andando a intaccare la tensione ai capi della capacità di uscita del buffer non lineare da 5pF.
Lo scopo è quindi quello di "isolare" la parte associata alle capacità C1 e C2 del BLH (vedi figura 2.9) nella fase in cui lo Shaper ritorna al suo stato iniziale. Si è adottata una soluzione che di fatto sfrutta il principio del Sample and Hold che ha evidenziato la possibilità di dimezzare il tempo di inhibit raggiungendo i 12/13µs.

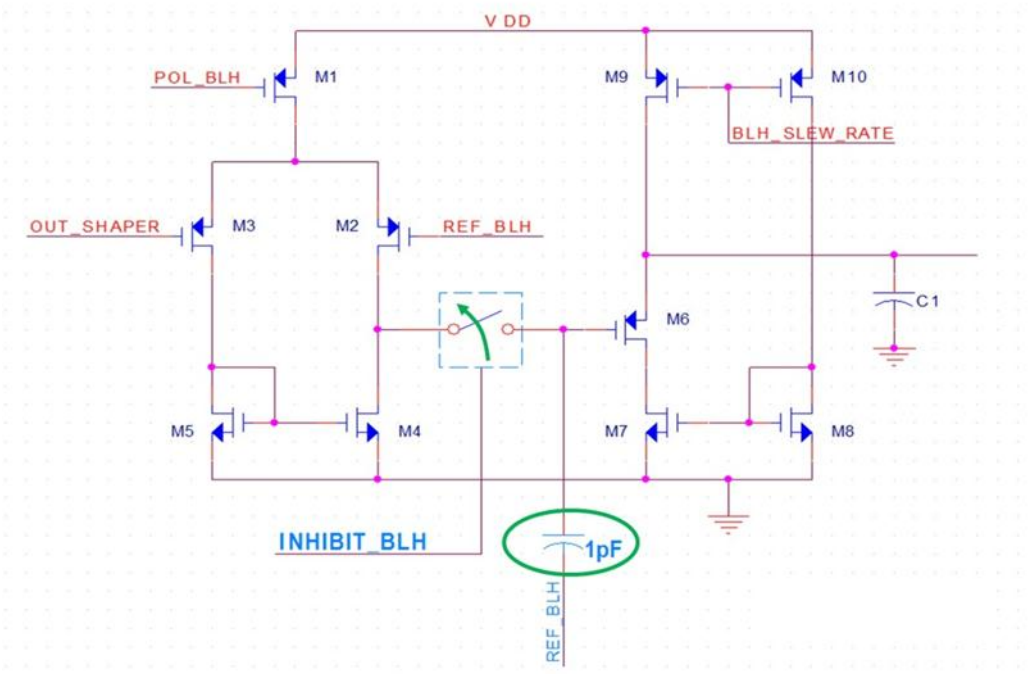


Figura 2-9 Dettaglio della soluzione pensata per l'inhibit del BLH

In ognuno dei 27 canali che costituiscono il singolo ASIC, è distribuita della logica che permette di eseguire il kill del canale stesso; le ragioni che spingono a voler inibire del tutto una o più linee analogiche sono sostanzialmente due:

il canale da inibire è associato a uno di quei 12 SDD della matrice da 81, che non rientrano nella finestra circolare di uscita dello scintillatore, come mostrato in figura 2.10;
il canale in fase di testing degli ASIC presenta un rumore notevole ed è inutile acquisirne l'informazione d'uscita.

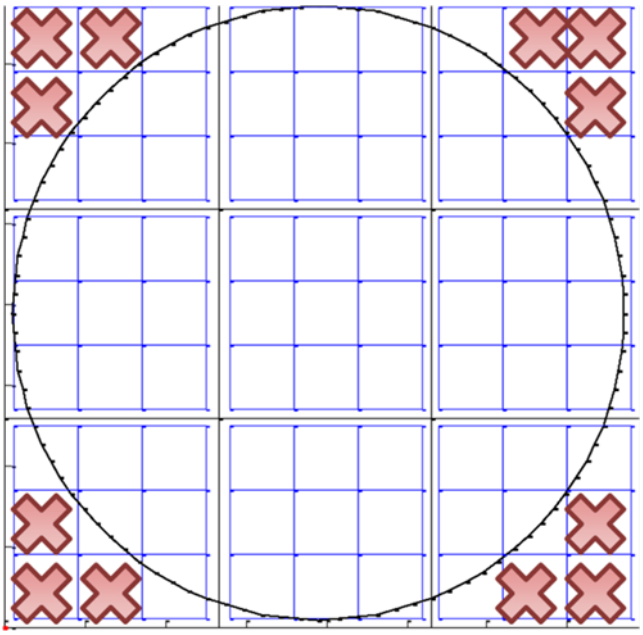


Figura 2-10 Dettaglio dei 12 SDD killati nella matrice perchè esclusi dalla finestra circolare dello scintillatore

In presenza di canali più rumorosi, infatti, risulta utile alzare la soglia di scatto del trigger associato alla rivelazione di un segnale utile. Per fare in modo che il numero di bit necessari per impostare questa soglia non fosse eccessivo si è deciso di creare una soglia minima comune a tutti i canali regolabile dai 6 bit del registro di programmazione che può essere regolata in modo più fine da ulteriori 3 bit del registro. Qualora però questi tre bit siano settati tutti a uno il canale in questione risulta inibito.

Il Peak Stretcher

Qualsiasi sia la forma della risposta all'impulso del filtro formatore adottato, è bene sottolineare che nel caso in cui si voglia ricostruire l'energia complessiva dei fotoni che incidono sul detector è necessario recuperare una sola informazione di tale impulso: la sua ampiezza; ed è proprio questa la specifica funzione del Peak Stretcher, ricevere in ingresso l'uscita del filtro formatore, "seguirne" l'andamento fino al

picco e mantenerne poi costante il valore di picco, in modo da garantire agli stadi successivi di elaborare l'informazione resa. Il circuito più semplice che può essere pensato, in grado di svolgere la funzione sopra descritta, è banalmente rappresentato dalla serie di un diodo e di una capacità;

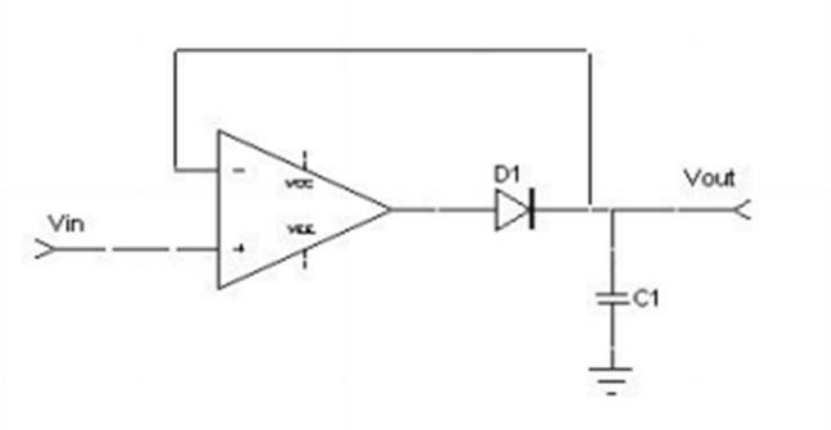


Figura 2-11 Il classico allungature di picco

E' poi presente una sezione logica di controllo per la gestione dei segnali di trigger, per il killing del canale e per la regolazione delle soglie.

Il Multiplexer Analogico

All'interno del chip ESA è presente un multiplexer analogico che serve per condividere una singola interfaccia verso la scheda di acquisizione. Questo è importante perché il DAQ può essere posto lontano dalla scheda degli ASIC e si richiede un buffer che funga da driver per il cavo. La condivisione di questo buffer tra più canali riduce il consumo di potenza e semplifica il layout su scheda. Inoltre, utilizzando il multiplexer analogico integrato all'interno del ASIC, è possibile condividere un singolo ADC senza ulteriori componenti.

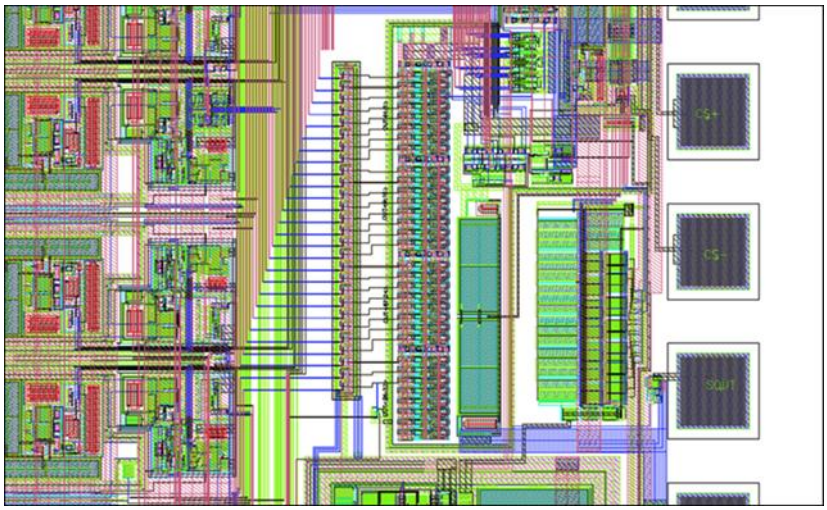


Figura 2-12 Al centro il Layout associato al Multiplexer e i suoi registri dedicati

Questo Multiplexer a 27 canali connette le uscite dei 27 Peak Stretcher ciascuna di esse prese singolarmente all'ingresso del buffer d'uscita per un lasso temporale di 100ns (se a frequenza di 10MHz), in modo da racchiudere in 2.7µs complessivi la "stringa" di ampiezze che caratterizza il singolo evento.

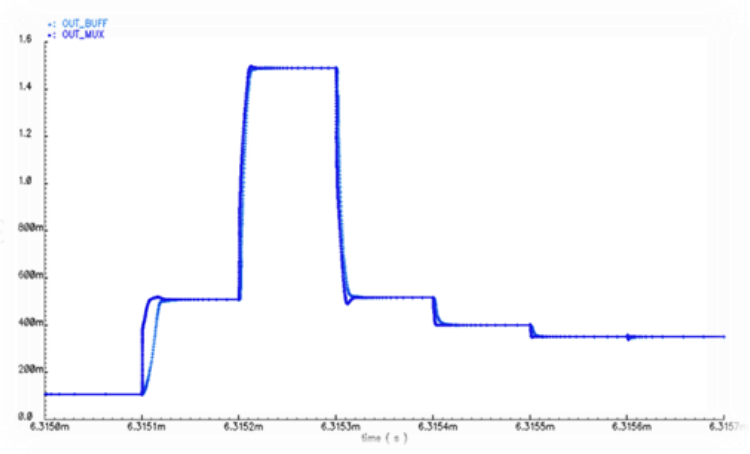


Figura 2-13 Segnale all'uscita del MUX e del buffer d'uscita a confronto

Capitolo 3

Hardware e firmware della sezione di Acquisizione

In questo capitolo viene presentata l'elettronica realizzata per l'acquisizione dei dati e la comunicazione dal PC alla sezione di rivelazione.
I segnali in uscita dai tre ASIC vengono inviati al sistema DAQ dopo l'opportuno condizionamento prima descritto. I 3 flussi di dati vengono campionati in parallelo e sono quindi inviati al processore digitale. Il processore è basato su un dispositivo logico programmabile di tipo FPGA che viene utilizzato al fine di effettuare l'elaborazione digitale richiesta e il controllo dei 3 ASIC. Un microprocessore ausiliario viene utilizzato per gestire l'interfaccia di comunicazione Ethernet con il PC. La scelta di un FPGA per la gestione dei dati al posto di un dispositivo a microprocessore, come ad esempio un dispositivo DSP, deriva dal fatto che un tale dispositivo è capace di gestire tanti segnali contemporaneamente in modo molto veloce, requisito indispensabile di questo sistema. Gli ulteriori vantaggi di un FPGA sono ampiamente descritti nel capitolo 5.

Architettura Hardware

L'unità di elaborazione digitale è fisicamente partizionato in 5 schede: le 3 schede per la conversione A-to-D, una scheda madre di elaborazione e una scheda di comunicazione.

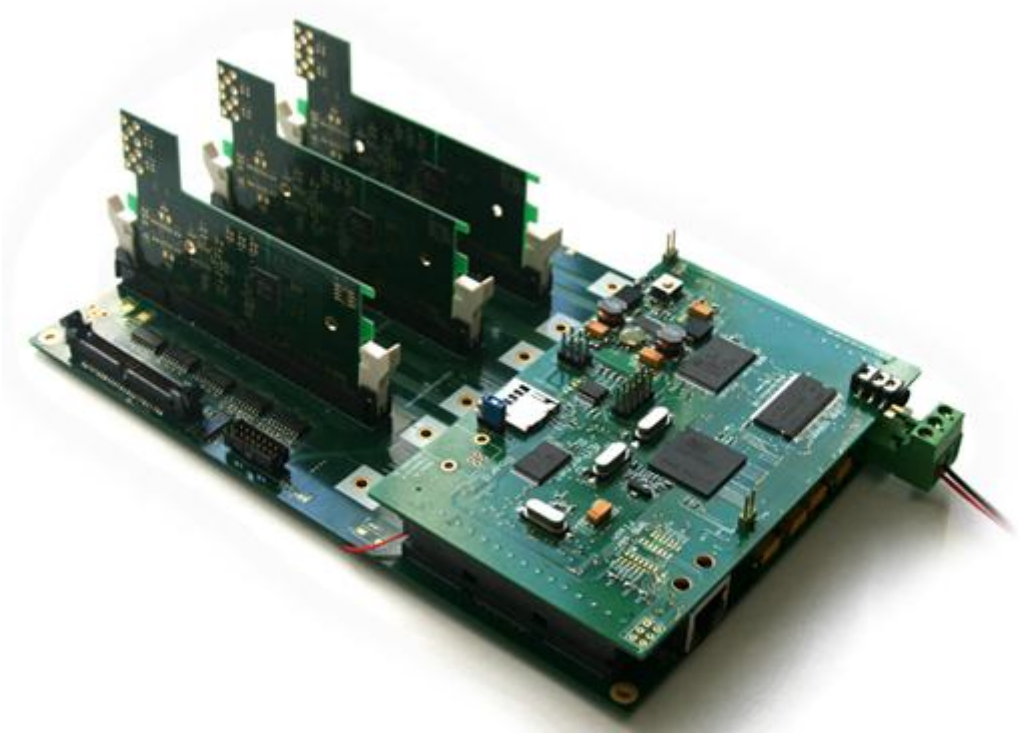


Figura 3-1 Scheda DAQ assemblata

Ogni scheda A-to-D è provvista di circuiti analogici di condizionamento del segnale, di un DAC per la regolazione dell'offset in ingresso all'amplificatore e di un ADC differenziale per ognuno dei 3 canali.

Figura 3-2 Scheda analogica

La scheda madre ospita le schede A-to-D e la scheda di comunicazione che sono alloggiati come moduli snap-in. La scheda

madre fornisce l'alimentazione analogica e digitale a tutto il sistema e con il dispositivo FPGA genera i segnali di temporizzazione necessari all'ASIC e al fine di sincronizzarlo con il processo di campionamento. L'FPGA gestisce inoltre il processo di conversione e genera i pacchetti di dati da inviare al PC.
La scheda di comunicazione è basata su microprocessore Atmel. Essa riceve il frame dati dalla scheda madre e li invia al PC attraverso un'interfaccia ethernet.
Il PC riceve i dati grezzi, li elabora e ricostruisce l'immagine usando la tecnica del baricentro.

La partizione del sistema in sottosistemi ha diversi vantaggi:

- Riduzione del rumore. Un sistema costituito sia da segnali analogici che digitali richiede alcune accortezze nella progettazione per ridurre al minimo il rumore di accoppiamento tra i circuiti digitali e quelli analogici Collocare la parte analogica distante da quella digitale aiuta a ridurre tale problema. In effetti, una separazione di almeno 3cm tra un circuito digitale con segnali ad elevata velocità e un circuito analogico, che subisce gli effetti di quello digitale, può ridurre il crosstalk di più di 60 dB.
- Dissipazione di potenza. Il front-end analogico e l'ADC richiedono che l'alimentatore sia in grado di erogare più di 25 W di potenza. Posizionare l'unità di elaborazione digitale sulla stessa board della sezione analogica porterebbe al surriscaldamento dell'intero sistema.
- Affidabilità. Il front-end analogico è direttamente collegato al ASIC e alla matrice di SDD che è polarizzata con più di 200 V: nel caso di sovratensione, solo la scheda analogica di front-end verrebbe danneggiata, lasciando intatta quella digitale (nettamente più costosa).
- Minimizzazione dei costi. La scheda madre richiede uno specifico e molto costoso processo di produzione del circuito stampato. Al contrario, le altre due schede sono realizzate con processi standard.
- Interfaccia di comunicazione. L'interfaccia standard di comunicazione tra PC e il sistema non è stabilita a priori. La scheda Ethernet potrebbe essere sostituita, per esempio, con un controller USB o un driver per fibra ottica.

La carica dai rivelatori viene convertita in una tensione differenziale che viene campionata da un ADC posto sulla scheda analogica. La

scheda di conversione è controllata dalla scheda madre per garantire che il multiplexer di uscita dell'ASIC generi il corretto valore analogico all'istante di tempo in cui il segnale viene campionato dall'ADC.
La scheda madre riceve i 3 flussi paralleli e li memorizza in una memoria per generare l'immagine 9x9. L'immagine viene poi serializzata e trasmessa alla scheda Ethernet.
Qui, un driver legge i dati dall'FPGA come pacchetti di 100 frames alla volta. Ciascun frame contiene o valori della matrice 9x9. Se vengono selezionati meno rilevatori, le uscite relative ai rilevatori non selezionati sono poste a zero. A livello software è stato implementato un driver che viene inserito nel kernel Linux in esecuzione sul dispositivo AP7000. Esso invia i dati ad una FIFO software. Dal lato opposto un server TCP attende la connessione di un PC al fine di mettere in comunicazione il PC con la FIFO stessa.
Il PC a sua volta scarica il flusso dati dal server Ethernet utilizzando una DLL personalizzata. I frame dati sono quindi passati al un software di elaborazione.

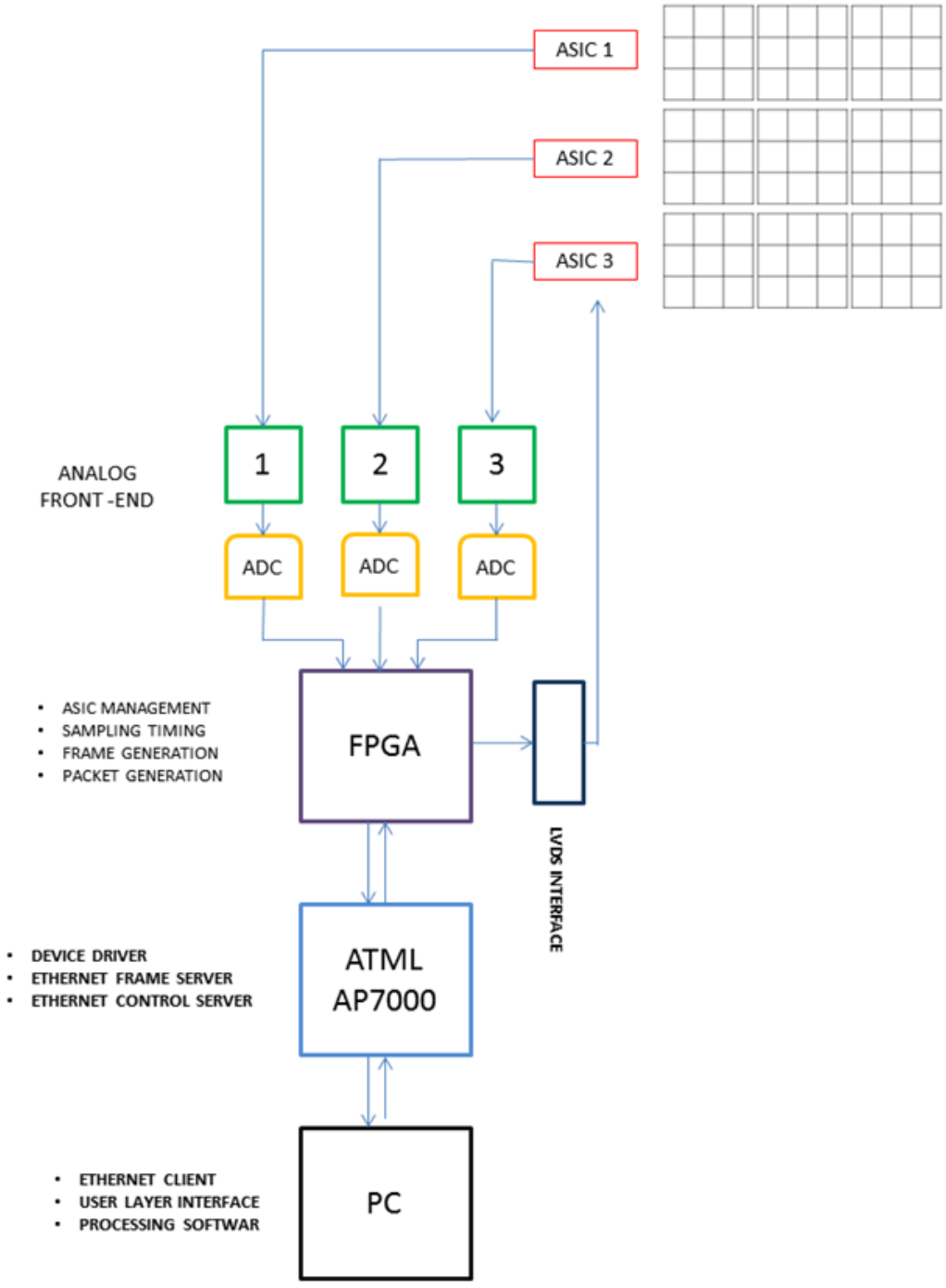


Figura 3-3 Schema a blocchi dell'intero sistema DAQ

Scheda di conversione analogico-digitale

L'architettura della scheda di conversione A-to-D è mostrata nella figura.

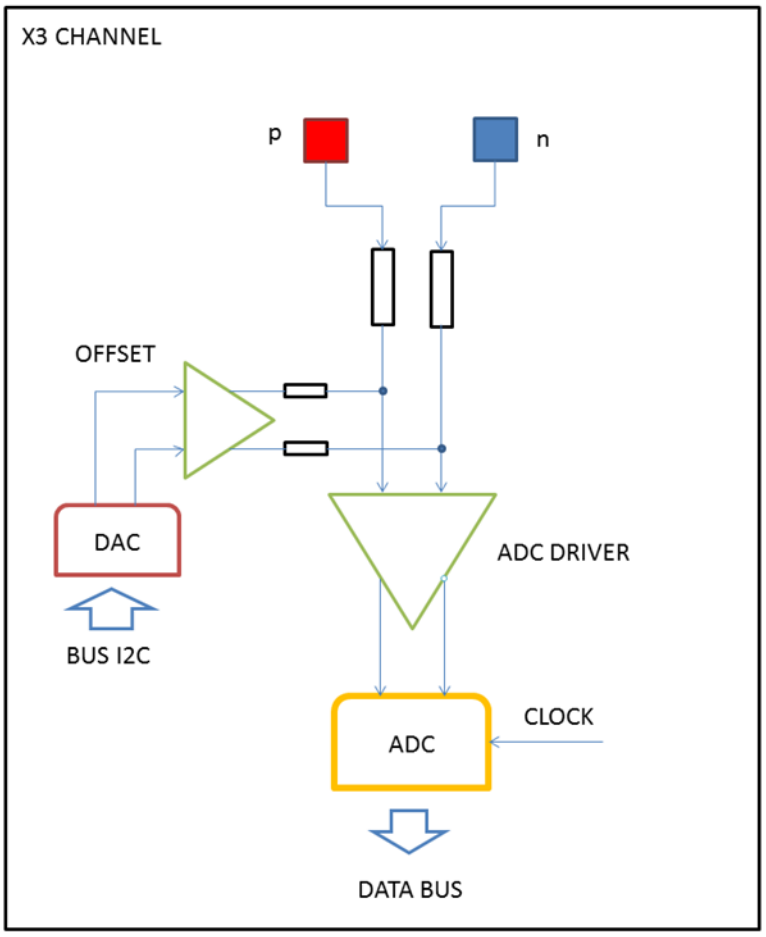


Figura 3-4 Schema a blocchi scheda analogica

L'ADC è un LTC2215 a 16 bit e può operare fino a 80 MSPS. Esso ha un ingresso completamente differenziale con una dinamica di 3.3V. L'ADC viene utilizzato in modalità free-running e l'uscita del ASIC è sincronizzato con l'istante di campionamento del convertitore.
Sono presenti tre canali di input completamente differenziali. Il primo stadio introduce un guadagno di 2 sul percorso del segnale e limita la larghezza di banda a 30 MHz al fine di ridurre il rumore. Un amplificatore fully differential, l'AD8139 di Analog Devices, è stato utilizzato per pilotare l'ADC e regolare l'offset. Ciò è necessario in quanto il segnale generato da ogni ASIC ha una polarizzazione spostata verso una delle alimentazioni. Ciascuno dei 3 canali dispone

di una propria indipendente regolazione dell'offset, impostabile digitalmente attraverso un DAC, l'AD5667R di Analog Devices, comandato dall'FPGA tramite un BUS I2C. Il pin VOCM del AD8139 è pilotato dal riferimento di modo comune di tensione generato dall'ADC. L'LTC2215 è un convertitore a CMOS a 7 stadi di pipeline. Ogni stadio, eccetto gli ultimi due, ha un ADC connesso a un DAC di ricostruzione e a un amplificatore. L'amplificatore amplifica la differenza tra il segnale ricostruito dal DAC e l'input dello stadio successivo. Il layout della scheda è mostrato nella Figura 3-5, in essa sono stati rigorosamente implementati tutti i requisiti di progettazione di un sistema misto analogico-digitale...da continuare

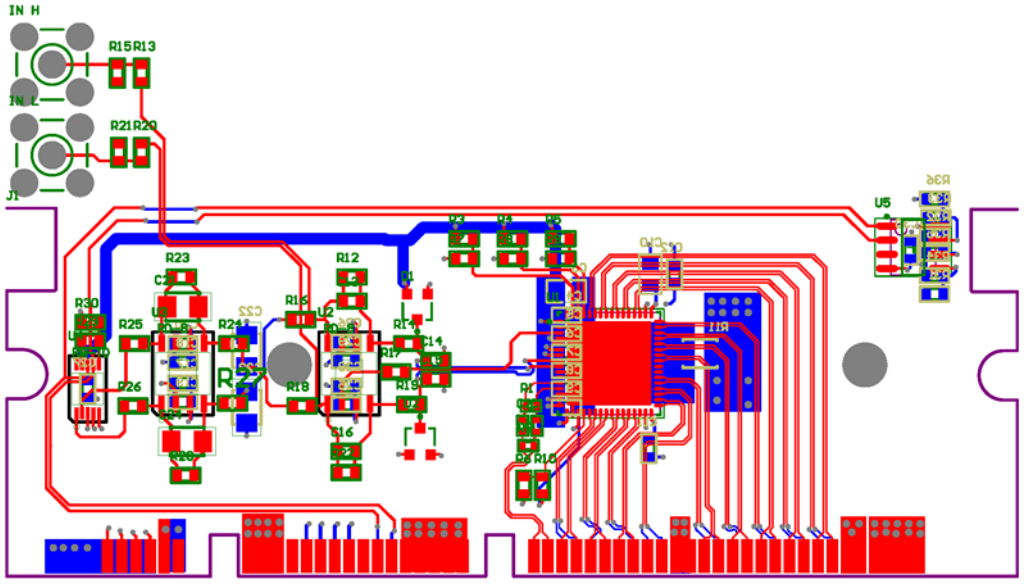


Figura 3-5 Layout scheda analogica

Hardware scheda madre

L'architettura della scheda madre è mostrata nella figura 3.6.
Sulla scheda sono saldati 2 connettori per l'alloggio della scheda ethernet, un connettore per la comunicazione con gli ASIC e 3 snap-in socket per le tre schede analogiche che ospitano l'ADC. Le tensioni per alimentare gli ADC sono fornite da tre regolatori di tensione LDO alimentati dall'esterno con una tensione di +5V. E' stato deciso di utilizzare generatori di tensione distinti al fine di distribuire la dissipazione di calore su una superficie maggiore. I regolatori positivi

sono degli REG1117-5 della Linear Technology. Per massimizzare la reiezione del rumore HF, sono stati introdotti adeguati filtri LC sia sull'ingresso che sull'uscita dei regolatori. Le tensioni esterne +5/-5V alimentano gli OP AMP della scheda analogica. L'alimentazione del sistema digitale è fornita da tre alimentatori switching che convertono i 12V esterni nelle seguenti tensioni: 1.2 V per il core FPGA, 3.3 V per tutta la logica e per il dispositivo di memoria flash. Un regolatore lineare è usato per generare la tensione di 2.5 V VCCAUX per i PLL interni al dispositivo FPGA.

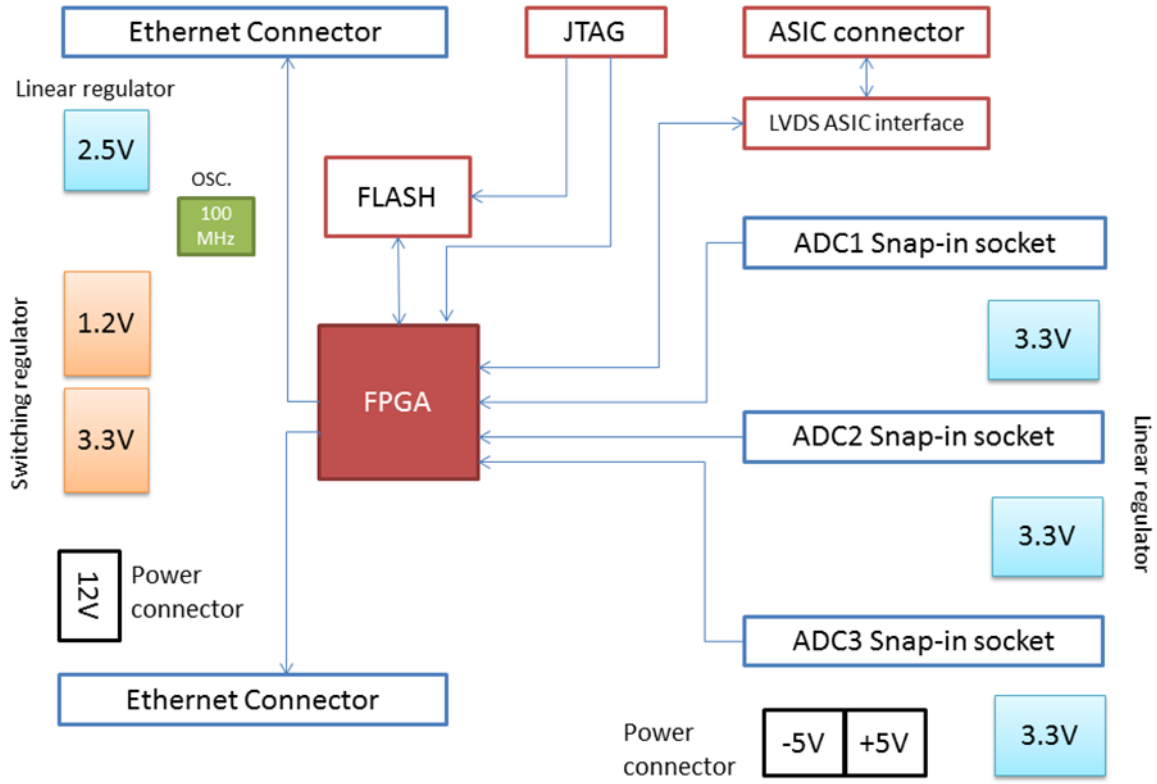


Figura 3-6 Architettura scheda madre

Il bus tra ADC e FPGA è costituito da 16 bit di dati più 1 segnale di clock tutti LVDS per ogni ADC. Il dispositivo FPGA è una Spartan-3A DSP della Xilinx, che risulta essere la migliore soluzione per l'applicazione di riferimento. Essa fornisce risorse logiche sufficienti e blocchi di elaborazione per effettuare tutte le operazioni richieste dalle specifiche di progetto. Inoltre, tale dispositivo permette di ampliare le funzionalità implementate in quanto circa il 50% delle risorse FPGA sono ancora disponibili. Il dispositivo è un ottimo

compromesso tra la dissipazione di potenza, costi e risorse. La configurazione del dispositivo FPGA è memorizzata in una memoria Platform Flash PROM XCF32P ,sempre della Xilinx, programmabile con il tool di progettazione tramite bus JTAG.
La comunicazione con gli ASICs avviene con un interfaccia LVDS esterna, collegata al connettore, formata da 5 driver SN65LVDS391d della Texas Instruments che convertono i segnali single-ended dell'FPGA in segnali differenziali verso gli ASICs. Questa è stata una scelta progettuale: l'FPGA conterrebbe un numero di pin di I/O sufficienti a ospitare tutti i segnali LVDS del sistema, gli ADC sono infatti collegati ad essi. Nella comunicazione con gli ASICs la scheda si interfaccia con un sistema esterno e suscettibile di qualsiasi problema tecnico (cortocircuiti, sovratensioni o errori nello standard di comunicazione), al fine di prevenire guasti dell'FPGA si è scelto di utilizzare dei driver esterni come schermo per l'FPGA. I segnali di controllo dell'ASIC operano a velocità di 10 Mbit/s, ma la distanza tra l'unità di elaborazione digitale e l'ASIC è di circa 40cm. I due dispositivi sono collegati con un cavo piatto non schermato. Per questi motivi i segnali utilizzano il protocollo LVDS al fine di garantire una migliore integrità del segnale e per ridurre il crosstalk tra le linee.

Non ci sono particolari problemi di signal integrity per quanto riguarda la comunicazione sul bus digitale tra ADC e FPGA: il segnale più veloce è quello di clock che presenta una frequenza massima di 40 MHz. Il crosstalk, il matching di lunghezza e terminazione delle linee non sono critici: la lunghezza massima della linea di trasmissione da pad a pad è 5cm. Basti considerare un fronte di salita di durata pari 1 / 10 del periodo di clock: la lunghezza spaziale del segnale è di 60cm, che è più di sei volte superiore alla lunghezza fisica del circuito. Inoltre, in questa gamma di frequenze di funzionamento il contributo del crosstalk è trascurabile.

Il dispositivo FPGA è direttamente collegato alla scheda Ethernet ed è mappata come una memoria statica per mezzo di un bus indirizzi a 26 bit e un bus dati a 16 bit.
Il microprocessore Atmel può accedere direttamente ai registri di configurazione del dispositivo FPGA attraverso un bus dedicato SPI, vale a dire un protocollo molto semplice che utilizza solo quattro linee di comunicazione. L'utilizzo di un bus secondario per la configurazione permette la comunicazione senza interferire con il

processo di scambio dei dati. Il bus SPI è utilizzato alla frequenza di 16 MHz.
Il bus dati a 16 bit viene fatto funzionare alla frequenza di 50 MHz; esso corrisponde a una velocità di trasferimento di circa 1,6 Gbit/s che è più di dieci volte la velocità del bus Ethernet.
La sorgente di clock per il dispositivo FPGA è uno cristallo standard con frequenza di 100 MHz. Il clock dei convertitori ADC viene generato internamente dal dispositivo FPGA. L'uscita dell'ASIC è costituita da una serie di 27 gradini di tensione indipendenti, uno per ogni SDD. Da questo punto di vista, né il teorema di Shannon, né tutte le considerazione sul jitter devono essere prese in considerazione. Il jitter nel caso di una misura singola non è un problema perché possiamo considerare il segnale di uscita del ASIC come valore DC stabile per tutto il tempo della conversione.
La scheda è progettata su 6 layer, il cui layout è mostrato nella figura 3.7.

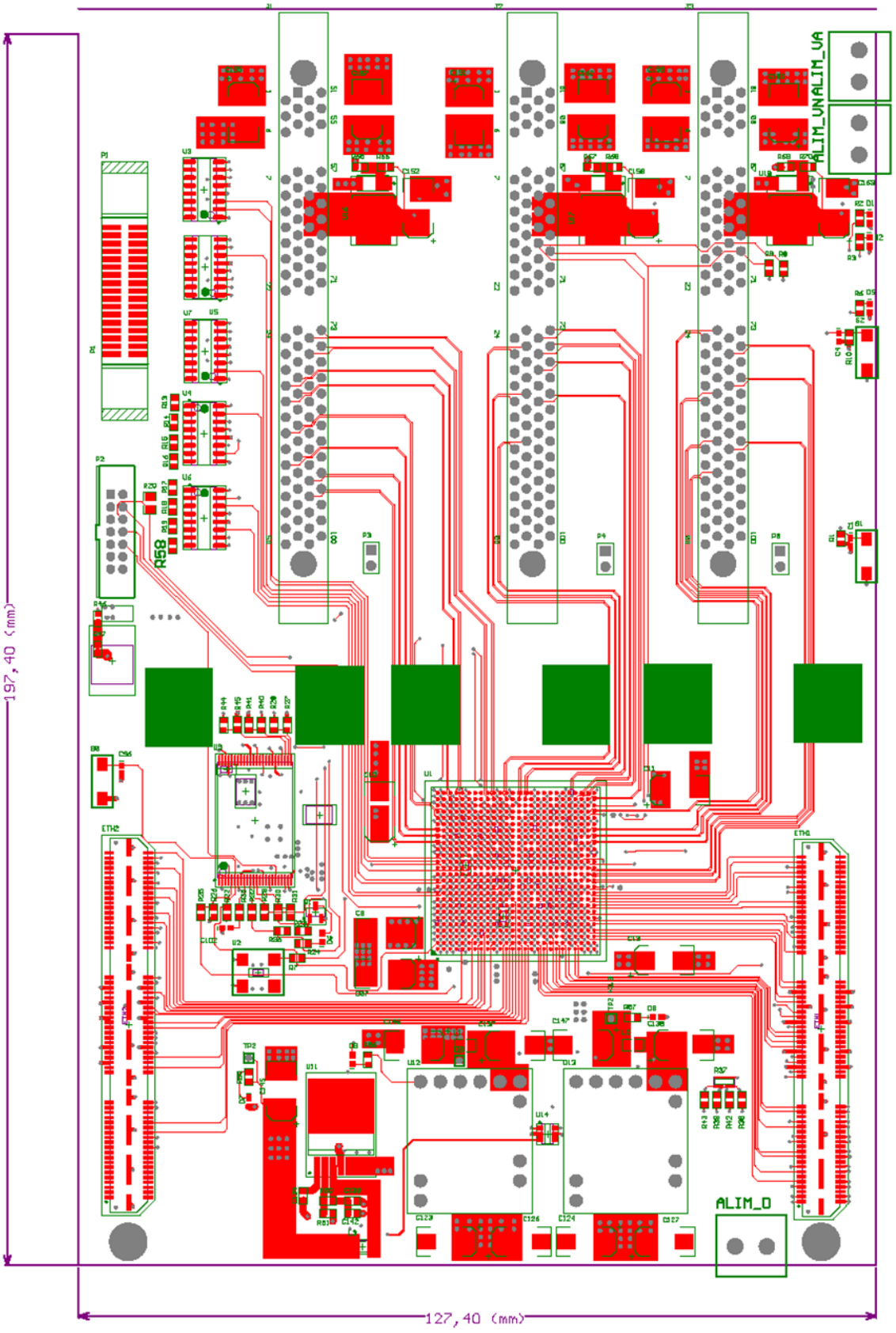


Figura 3-7 Layout scheda madre

lo stackup della scheda è invece mostrato nella figura 6.

Figura 3-8 Stackup scheda madre

Su uno dei due layer di massa è presente la massa delle schede analogiche divisa dalla massa del sistema digitale. Le masse sono poi collegate in maniera opportuna in modo da attenuare le componenti ad alta frequenza generate dal sistema digitale.

Scheda Ethernet

L'architettura della scheda Ethernet è illustrata in figura 3.9.

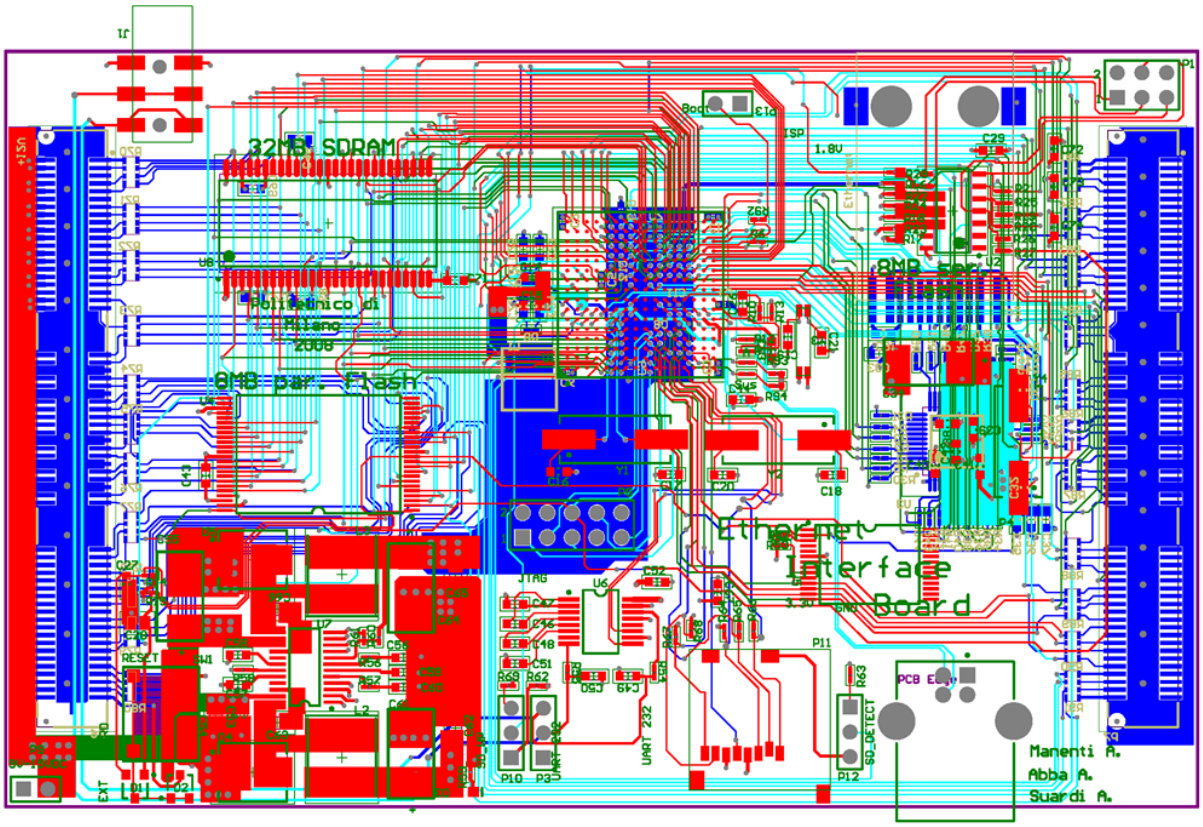


Figura 3-9 Layout scheda Ethernet

La scheda di rete è in realtà un SBPC equipaggiato con un processore in grado di eseguire una versione completa di Linux con kernel 2.6,

un'interfaccia 10/100 Ethernet, 32 Mbyte di SDRAM, due moduli di memoria Flash da 8 MB uno seriale e uno parallelo e una scheda SD per una capacità totale di memorizzazione totale pari a più di 1 Gbyte. Il compito principale della scheda Ethernet è quello di fornire due servizi Ethernet TCP: uno che trasmette il frame dati (81 valori di energia campionati dalla matrice SDD) e l'altra che controlla e inizializza i parametri di elaborazione e di configurazione dell'ASIC attraverso il dispositivo FPGA. Il cuore del sistema è il microprocessore Atmel AP7000. Si tratta di un microprocessore a 32 bit ARM9 con MMU, clock a 250 MHz che include interfacce I / O come SPI, EBI, I2C, MII (specifico per l'interfaccia Ethernet), RS232. Il bus EBI è utilizzato per connettere al processore l'FPGA e i dispositivi di memoria, ovvero la SDRAM e le memorie parallele Flash. Il bus EBI utilizza una logica CMOS 3,3 V e opera in single data rate. A causa del funzionamento ad alta velocità del bus EBI (133 MHz quando comunica con l'AP7000 SDRAM, 50 MHz quando comunica con l'FPGA) gli effetti di cross-talk, riflessioni e ritardo di propagazione devono essere considerati. Il bus progettato ha una topologia a stella ed è diviso in due diversi livelli: uno collega in serie memoria SDRAM e Flash mentre l'altro il chip FPGA. Nel progetto della scheda è stato adottato lo stackup a 6 layers di figura 3.10.

Figura 3-10 Stackup scheda Ethernet

La porta MII è collegata ad un Transceiver per interfaccia Ethernet (DP83848 della National Semiconductor). L'AP7000 implementa lo stackup Ethernet fino al livello di accesso e controllo dei media (MAC). Al fine di interfacciarsi con il cavo Ethernet, è necessario avere un controllore esterno di livello fisico. Il bus MII (standardizzato da IEEE 802.3u) è un bus generico che collega i diversi tipi di PHY al livello Media Access Controller (MAC). I dati posti su bus MII sono trasferiti utilizzando parole di 4 bit (nibble) in entrambe le direzione, con clock a 25 MHz, raggiungendo così una velocità di 100 Mbit/s. La

connessione tra l' AP7000 e il cavo Ethernet è illustrato nella figura 8. I trasformatori sono integrati nei connettori Ethernet RJ-45.

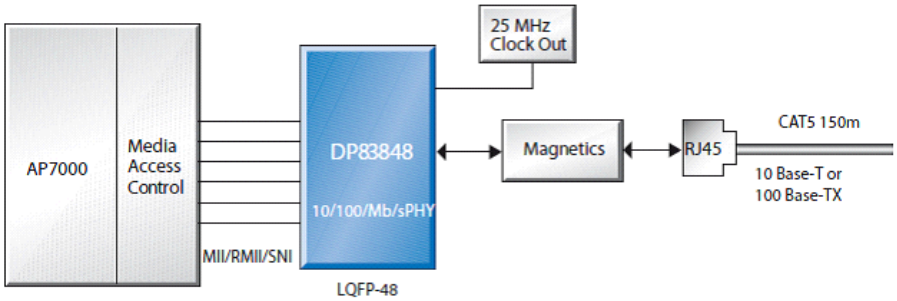


Figura 3-11 Collegamento tra AP7000, layer fisico e cavo LAN

E' stato anche collegato un connettore microSD alla porta MCI al fine di poter montare la partizione user del file system.
L'interfaccia RS-232 viene utilizzata per accedere alla console di comando Linux. Un dispositivo MAX3232 implementa l'interfaccia fisica tra le linee TX / RX del AP7000 con il cavo standard RS232.Il circuito ha bisogno di due sole alimentazioni. Una a 3,3 V e una 1,8 V. Queste sono generate da un regolatore switching a doppia uscita che funziona con una tensione di ingresso compresa tra i 7 V e 15 V. La tensione di 1.8 V è richiesta dal core dell' AP7000, mentre quella di 3,3 V da tutti gli altri circuiti integrati.

Il processore AP7000

L'AT32AP7000 è un system-on-chip che integra un processore con architettura RISC AVR32 che è in grado di raggiunge i 210 DMIPS operando con clock a 150 MHz. L'AVR32 è un microprocessore RISC a 32-bit ad alte prestazioni, progettato per applicazioni embedded a basso costo, bassi consumi, alta densità di codice e applicazioni ad elevate prestazioni. AT32AP7000 implementa una Memory Management Unit (MMU) e un controller di interrupt flessibile; è così in grado di supportare sistemi operativi di ultima generazione e sistemi operativi real-time. Il processore include anche un ricco insieme di istruzioni SIMD e DSP, appositamente progettate per applicazioni multimediali e di telecomunicazione.
AT32AP7000 incorpora anche una memoria SRAM on-chip per l'accesso veloce ai dati. Per applicazioni che richiedono memoria aggiuntiva, è presente il bus di comunicazione EBI che garantisce l'accesso a memoria esterne SRAM a 16-bit. Inoltre, un controller di memoria SDRAM fornisce accesso a memoria volatile off-chip, nonché

anche a tutte le memorie off-chip non volatili, quali Compact Flash, Multi Media Card (MMC), Secure Digital (SD) card, Smart Card, NAND Flash. Il controller Direct Memory Access (DMA), disponibile anche per tutte le periferiche seriali, permette il trasferimento di dati tra le memorie senza l'intervento del processore. Ciò riduce l'overhead del processore quando viene richiesto un trasferimento continuo di dati tra diversi moduli MCU.
E' dotato di tre timer identici a 16-bit, ogni timer può essere programmato per eseguire autonomamente una vasta gamma di funzioni, tra cui la misura di frequenza, conteggio di eventi, intervallo di misura, generazione di impulsi, tempi di ritardo e modulazione della larghezza degli impulsi.

Bus di comunicazione coi dispositivi ASIC

Il bus di comunicazione tra ASIC e FPGA permette all'utente di configurare l'ASIC. Le funzioni selezionabili sono ad esempio la regolazione del guadagno e del tempo di formatura, la sincronizzazione del processo di campionamento con il corretto canale d'uscita e l'esecuzione di azioni come il reset del rivelatore o del peak stretcher. I segnali che compongono questo bus di comunicazione sono illustrati nella figura 3.12.

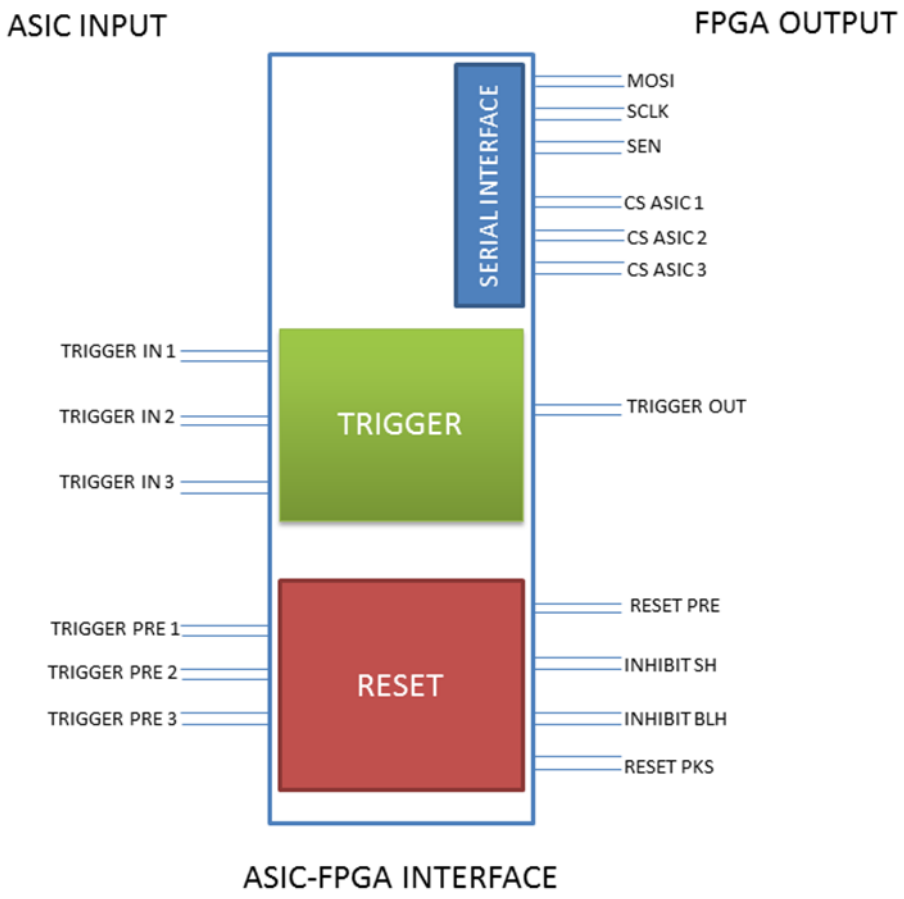


Figura 3-12 Schema dell'interfaccia tra DAQ e ASIC

Ci sono tre gruppi di segnali:

- Segnali di programmazione dell'ASIC;
- Segnali di trigger dell'ASIC;
- Segnali di reset;

Il gruppo di segnali ASIC di programmazione costituisce una versione LVDS di un bus unidirezionale SPI. Sono presenti due segnali, SCLK e MOSI, che rappresentano clock e bus dati e altri 3 segnali che agiscono come chip select per gli ASIC. Il dispositivo FPGA utilizza i segnali di programmazione dell'ASIC per due scopi: il primo, per programmare valori di guadagno, riferimenti di tensnione e correnti, disabilitare un canale e selezionare le uscita di monitor (fase di programmazione) e il secondo, per selezionare quale dei tre canali, ciascuno associato a 27 diversi peak stretcher, collegare all'uscita dell'ASIC affinchè venga campionato dal corretto ADC. Il segnale SEN (Serial Enable) seleziona se il bus seriale viene utilizzato come

interfaccia di programmazione o come segnale di ingresso al multiplexer di selezione.
Il gruppo di segnali di trigger è composto da 3 trigger-in e 1 trigger-out. Quando una certa quantità di fotoni colpisce un rivelatore, un segnale di trigger viene generato dall'ASIC che è connesso a quel determinato rivelatore e uno dei quattro trigger-in viene alzato. L'FPGA rileva l'evento di trigger e avvia il processo di acquisizione. Il livello di soglia è fissato per tutti e tre gli ASIC e ogni ASIC ha un proprio segnale di trigger. Per catturare l'intera area di rilevamento, quando l'FPGA rileva un trigger su uno dei 3 trigger-in, genera un impulso sul trigger-out il quale segnale è connesso in parallelo al circuito di innesco degli ASIC. L'effetto di un impulso sul trigger-out è la generazione di un trigger artificiale in tutti i dispositivo ASIC.

Timing del dispositivo

Il dispositivo ASIC può operare in tre differenti stati: programmazione del registro di configurazione, acquisizione, reset. In questo paragrafo verrà spiegato come il FPGA controlla i segnali degli ASIC nelle tre diverse modalità di esecuzione.

Programmazione del registro ASIC

All'interno dell'ASIC vi è un solo registro a 160 bit che contiene tutti i parametri di configurazione. I primi 41 bit permettono di impostare i riferimenti di tensione e corrente dei vari blocchi funzionali della catena di front-end. I successivi 6 bit regolano la tensione di soglia di comparazione all'uscita dello Shaper, tale regolazione definisce l'ampiezza minima necessaria tale da discriminare la presenza di un evento utile sulla matrice di SDD e generare un trigger sulla scheda d'acquisizione. Un bit è utilizzato per la funzione di auto-trigger. Se abilitato, a fronte di un evento, l'ASIC stesso comanda a tutti gli altri peak stretcher di leggere senza l'ausilio del segnale proveniente dalla scheda di acquisizione (trigger-out). Dopo 3 bit non utilizzati seguono 11 bit per la regolazione del guadagno dello shaper e del peaking time. Altri 81 bit permettono regolare finemente la soglia di comparazione all'uscita dello Shaper, 3 bit per ognuno dei 27 canali. Con 5 bit si può decidere di osservare in fase di testing una qualunque delle 27 uscite dello shaper e del preamplificatore; tutti i codici compresi fra 11011 e 11111 non sono utilizzati e implicano una connessione fra le uscite di test e il riferimento di tensione del

Baseline Holder. 4 bit consentono di abilitare/disabilitare le uscite di testing analogiche e digitali. L'ultimo bit permette di abilitare la funzione di doppia isteresi nel comparatore a soglia all'uscita dei 27 Shapers; ciò è utile quando l'obiettivo è quello di rivelare segnali molto piccoli.
Quando il pin SEN (linea verde) è impostato su 1, l'ASIC n-esimo selezionato dal piedino di chip select (CS, linea blu) entra in modalità di programmazione. Il registro di configurazione interno è azzerato e l'ASIC non è in grado di generare trigger. La parola a 160-bit di programmazione è generato dal software sul PC e trasmesso attraverso tutta la catena di comunicazione al controllore SPI nella FPGA. Il controllore SPI utilizza i segnali MOSI (linea rosa) e SCLK (linea gialla) per programmare correttamente la configurazione del registro ASIC. Al termine della fase di programmazione, il SEN è impostato a 0 e l'ASIC torna in modalità di acquisizione.

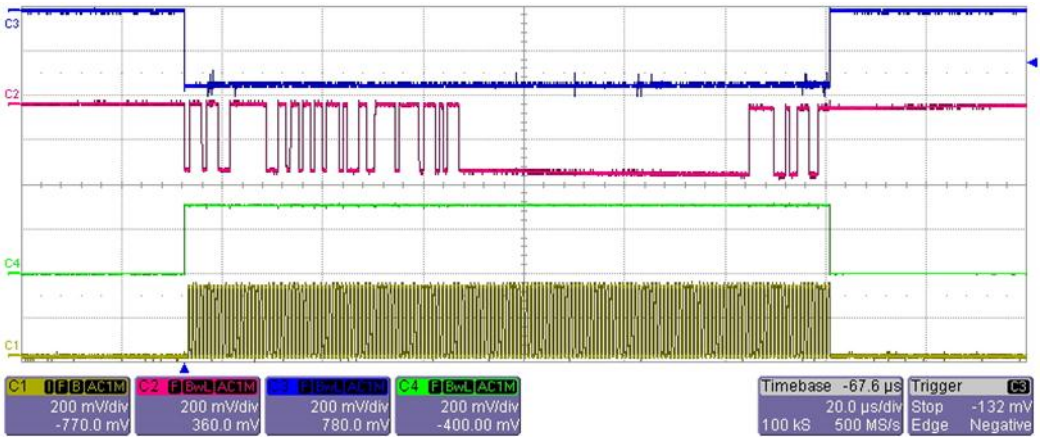


Figura 3-13 Schermata d'oscilloscopio che mostra l'andamento temporale dei segnali SPI in fase di programmazione.

La macchina stati riportata in figura 3.14 viene percorsa 3 volte ad ogni richiesta di programmazione, una per ogni n-esimo ChipSelect. Ad ogni fronte di salita dell'SCLK viene inviato sul SIN il bit i-esimo fino a raggiungere il 160-esimo. Dal lato ASIC il SIN viene letto sul fronte di discesa dell'SCLK, quando il dato è stabile.

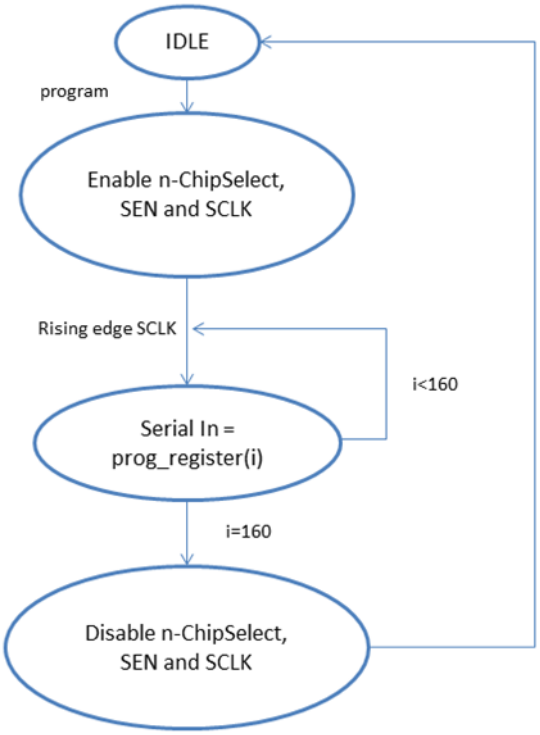


Figura 3-14Macchina a stati della fase di programmazione.

In figura 3.13 è stata mostrata la programmazione relativa al singolo ASIC. Questo perchè i 3 ASIC vengono programmati in finestre temporali differenti. Nella figura 3.15 sono rappresentati i tre segnali di ChipSelect, tra la programmazione di un chip e la programmazione del chip successivo intercorrono 200ms, tempo programmabile da PC.

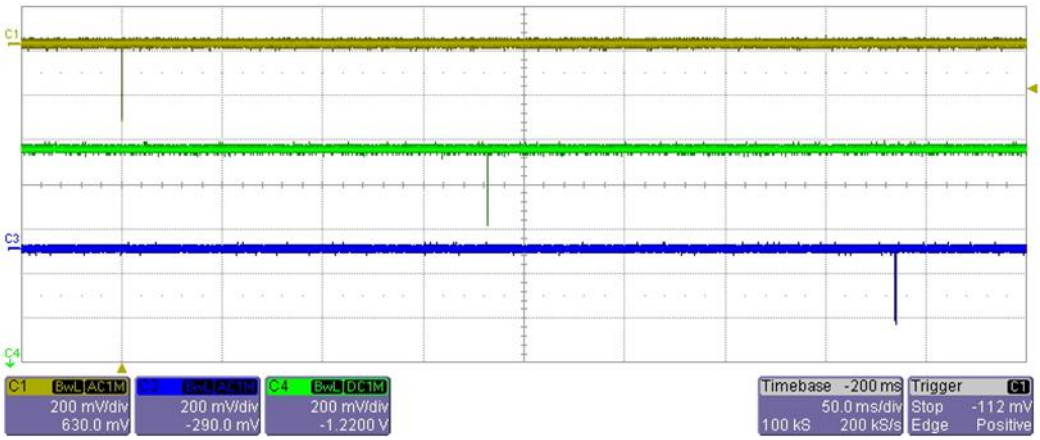


Figura 3-15 Programmazione dei tre ASIC

Reset del pre-amplificatore

La corrente di leakage del detector si integra nel tempo sulla capacità di feedback del pre-amplifitore, la tensione satura in pochi

millisecondi. La procedura di reset si occupa dunque di scaricare la carica la capacità e di inibire contemporaneamente lo shaper e la baseline holder per i motivi illustrati nel capitolo 2. La fase di reset è scatenata dai trigger-pre, generati da comparatori con soglia programmabile all'interno degli ASICs . I tre segnali di trigger sono posti in OR all'interno dell'FPGA e all'arrivo di uno dei tre vengono generati i segnali di reset nella corretta sequenza temporale come mostrato nella figura 3.16. I tre segnali sono posti a 1 tutti insieme e ritornano a zero con tempi diversi.

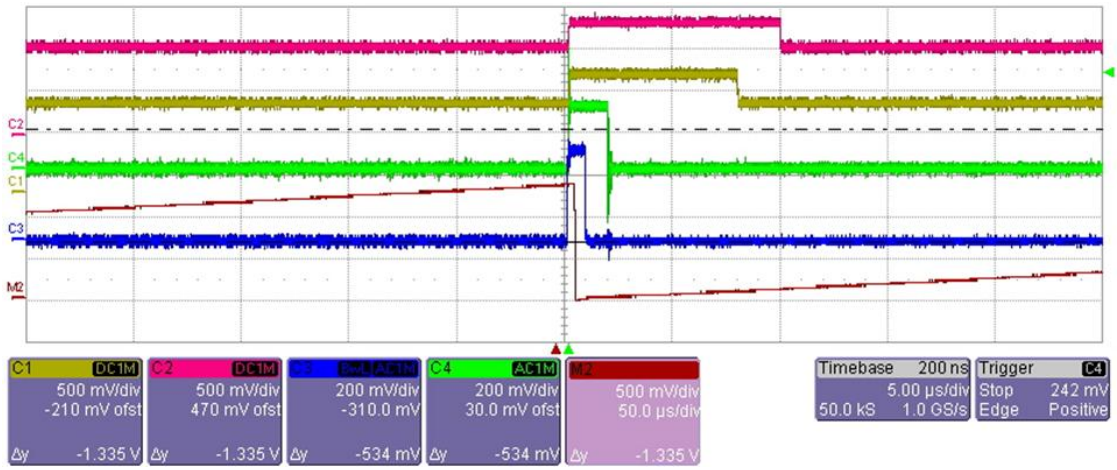


Figura 3-16 Segnali di reset

Nell'FPGA un segnale di busy mette in stallo un eventuale richiesta di programmazione fino all'esaurimento del processo di reset.

Temporizzazione dell'Acquisizione

Quando il fascio di luce colpisce uno o più rilevatori l'impulso di carica attraversa la catena di fronte-end e a valle dello shaper si genera una un segnale semigaussiano. Il segnale viene mandato a un comparatore con isteresi con soglia programmabile. L'uscita del comparatore genera il trigger d'acquisizione. Una volta che un ASIC attiva un trigger, il processo di acquisizione ha inizio.
Viene generato un segnale di trigger dall'FPGA che l'ASIC utilizza per attivare tutti gli altri peak stretcher. In queta fase i chip select di tutti e tre gli ASIC sono posti a zero abilitando cosi il multiplexer analogico nei tre chip. Nella figure in basso sono riportate le schermate della fase di test.

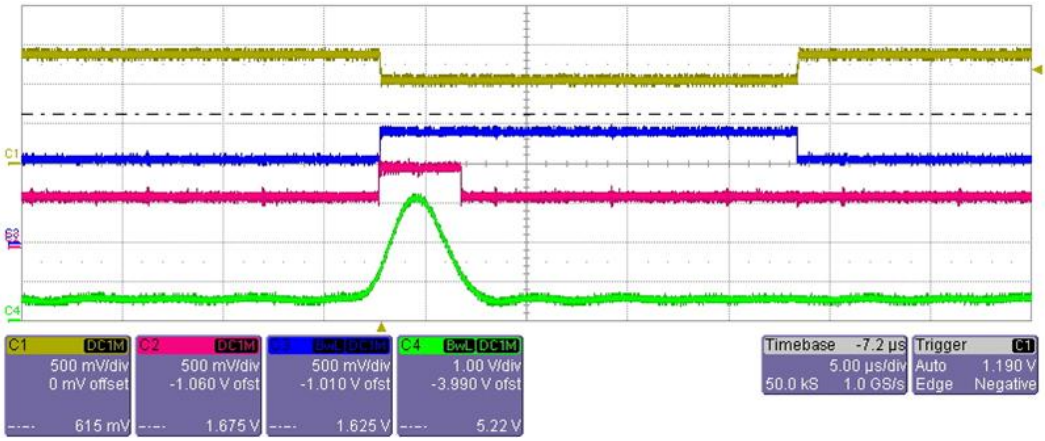


Figura 3-17 Dall'alto verso il basso: CS , TRIGGER_IN, TRIGGER_OUT e uscita dello shaper

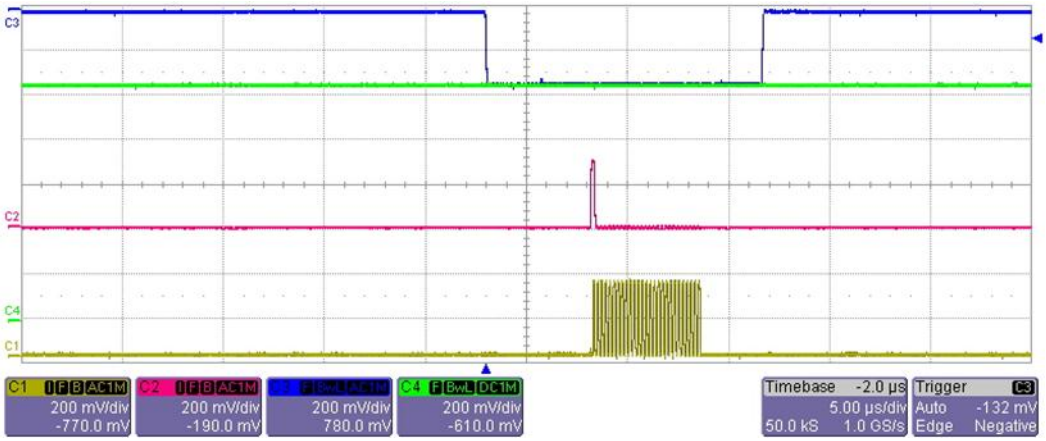


Figura 3-18 Dall'alto verso il basso: CS , SEN, MOSI e SCLK

Le uscite dei tre canali degli ASIC sono collegate alle schede analogiche alloggiate sulla scheda DAQ. Qui sono presenti gli ADC per la conversione.

E 'importante notare che l'ADC non può funzionare in modalità one-shot. Infatti, un ADC in pipeline ha bisogno di un clock che non si ferma mai, in quanto il tempo di recovery è pari a diversi periodi di clock.

Dal momento che l'ADC deve funzionare in modalità free-running, i segnali di clock dell'ASIC e dell'ADC devono essere adeguatamente sincronizzati. L'uscita dell'ASIC è una scala di tensioni analogiche. Ogni passo rappresenta la tensione di picco di un canale catturata dal corrispondente peak stretcher.

La figura 3.19 (a) mostra invece le forme d'onda ideale e reale.

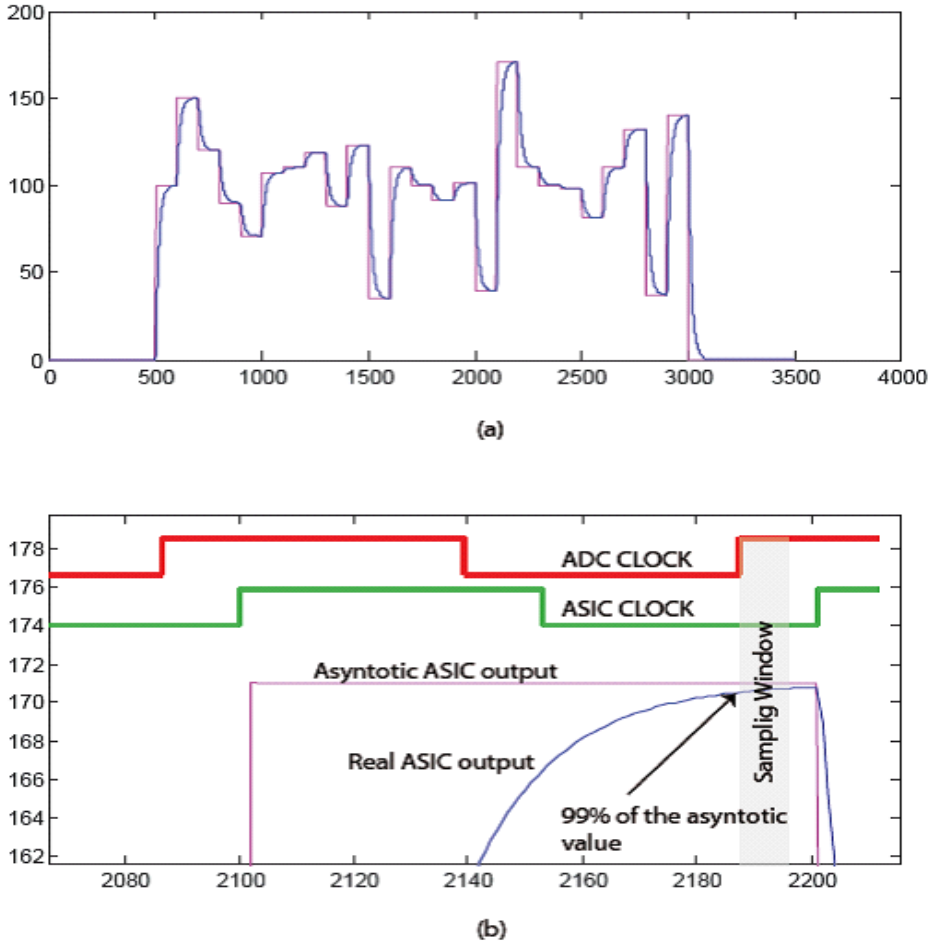


Figura 3-19 Forme d'onda in ingresso e segnali di clock

Quando il multiplexer dell'ASIC commuta, l'uscita di un peak stretcher viene collegata al buffer di uscita. Il buffer di uscita non è in grado di fornire elevate correnti e ciò non permette elevate velocità di commutazione. Pertanto le variazioni del segnale d'uscita seguono una curva logaritmica. È quindi necessario inserire un ritardo di fase tra il clock dell'ADC e il clock dell'ASIC come mostrato nella figura 3.19 (b).

Il processo di acquisizione inizia con un segnale di trigger che cambia lo stato del firmware nell'FPGA da attesa (IDLE) ad acquisizione. L'FPGA immediatamente alza il segnale di trigger-out e si auto inibisce in modo da non poter rilevare un ulteriore segnale di trigger. L'uscita del multiplexer viene selezionata scrivendo un 1 logico nello shift register dell'ASIC utilizzando le linee di MOSI e SCLK (con il segnale di controllo SEN posto a 0). Ad ogni periodo dell'SCLK, l'1 logico scritto nello shift register, viene spostato verso destra di una posizione, collegando così l'uscita del successivo peak stretcher all'uscita del ASIC. Quando la procedura di acquisizione è inizializzata, si deve attendere un tempo morto (T1) tra trigger e il

primo fronte di salita del clock per consentire al peak stretcher di accumulare correttamente la carica. Alla fine del ritardo T1, il primo peak stretcher viene campionato senza la necessità di un ulteriore fronte di clock in quanto l'1 logico nella prima posizione dello shift register viene caricato al termine del processo di acquisizione.

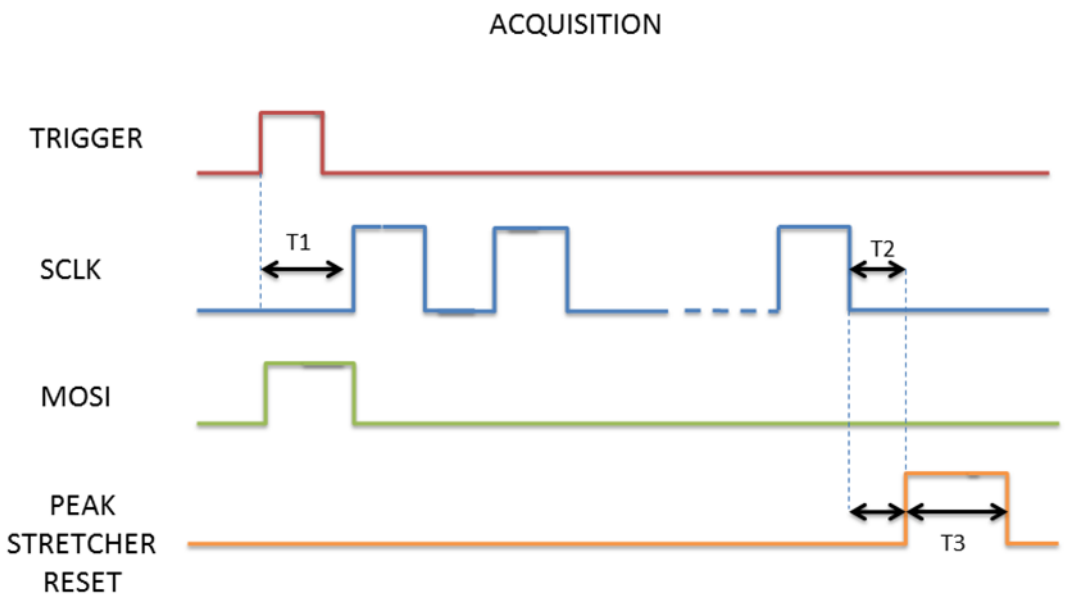


Figura 3-20 Timing d'Acquisizione

Dopo la prima acquisizione, per i 27 cicli di clock successivi avviene una connessione in sequenza dei 27 canali di ingresso del ASIC con l'ADC. L'ADC campiona la gradinata di segnali analogici in determinati istanti temporali. Il processo di acquiszione termina con un impulso di PEAK STRETCHER RESET che azzera il peak stretcher e lo shift register. Tutti i tempi T1, T2 e T3 sono programmabili da PC. Anche la frequenza dell'SCLK è programmabile, 2.5, 5 e 10 MHz. Gli ADC vengono pilotati con un clock 4 volte piu veloce dell'SCLK con uno sfasamento di 270 gradi per il motivo sopra spiegato. All'inizio del processo, lo shift register dell'ASIC è pre caricato con un 1 logico nella prima posizione, a ogni colpo di clock l'uno slitta per collegare il canale successivo all'uscita.
Dopo il reset del peak stretcher, il sistema di rilevamento è pronto a generare nuovi trigger. Riporto di seguito la macchina a stati del processo di acquisizione (figura 3.21)

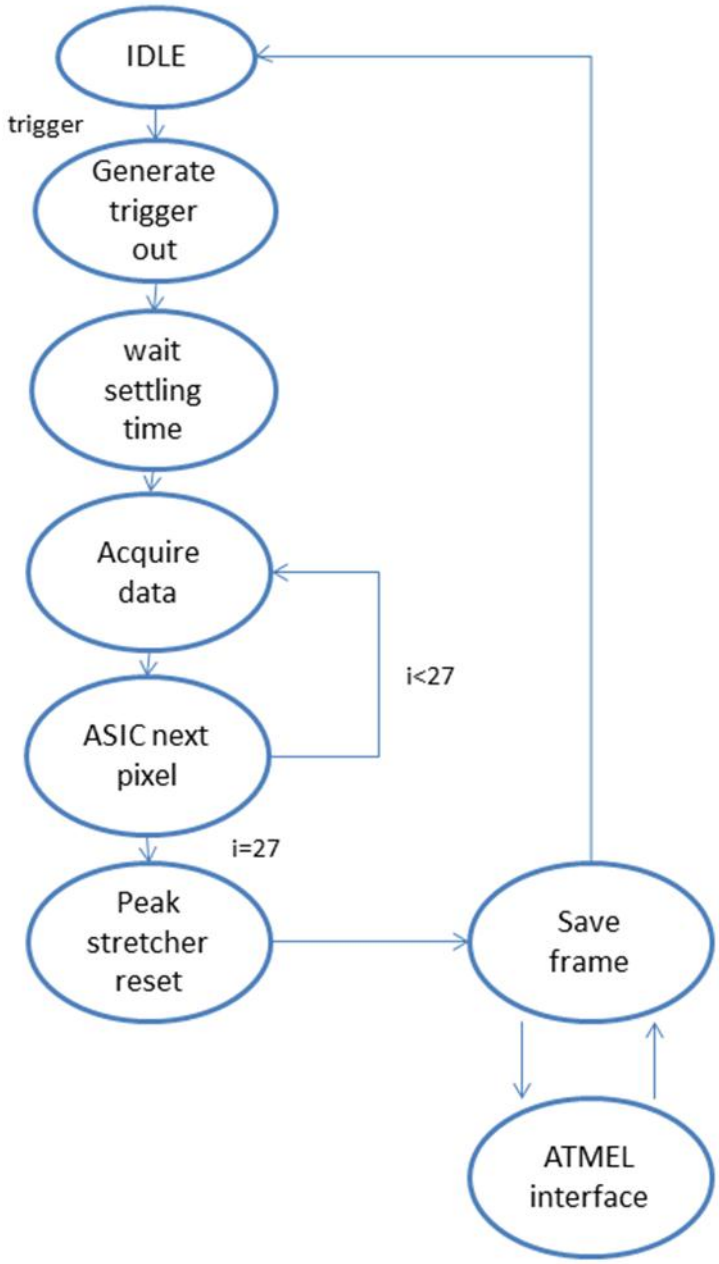


Figura 3-21 Macchina a stati Acquisizione

Ciascuna delle 3 schede funziona in parallelo. I segnali di controllo sono comuni ai 3 ASIC. Le uscita dei 3 circuiti di acquisizione (all'interno dell' FPGA) sono costituite da tre flussi di dati larghi 16 bit. Ogni flusso dati rappresenta i 27 SDD di un canale. Un blocco di elaborazione serializza i tre flussi di dati per generare il frame di 81 SDD. I frame vengono memorizzate in una BRAM a doppia porta che è accessibile da un lato dalle unità di elaborazione FPGA e dall'altro lato dal processore ATMEL.

Architettura firmaware del processore Atmel AP7000

La scheda di comunicazione Ethernet, è un Single Board PC ed è in grado di eseguire una distribuzione ad-hoc di Linux. È importante sottolineare che un vero kernel Linux 2.6 è in esecuzione sulla scheda Ethernet e non una versione uC-Linux. Ciò significa che il kernel usa l'MMU per separare lo spazio di memoria utente dallo spazio di memoria del kernel. La scheda Ethernet comunica con la scheda FPGA utilizzando sia il bus EBI che il bus SPI. È stato quindi necessario progettare un device driver personalizzato per interagire con l'hardware.
Il device driver viene caricato dal sistema operativo al momento dell'avvio.Lo user layer in questo progetto è costituito da due programmi che costituiscono due server Ethernet: il primo, chiamato "configuration server" e gestisce il bus SPI per impostare il file di registro della FPGA, il secondo, detto "server streamer frame", esegue lo scambio dei dati mediante il bus EBI. La struttura del firmware di comunicazione funzionante sulla scheda Ethernet è illustrato nella figura 3.22.

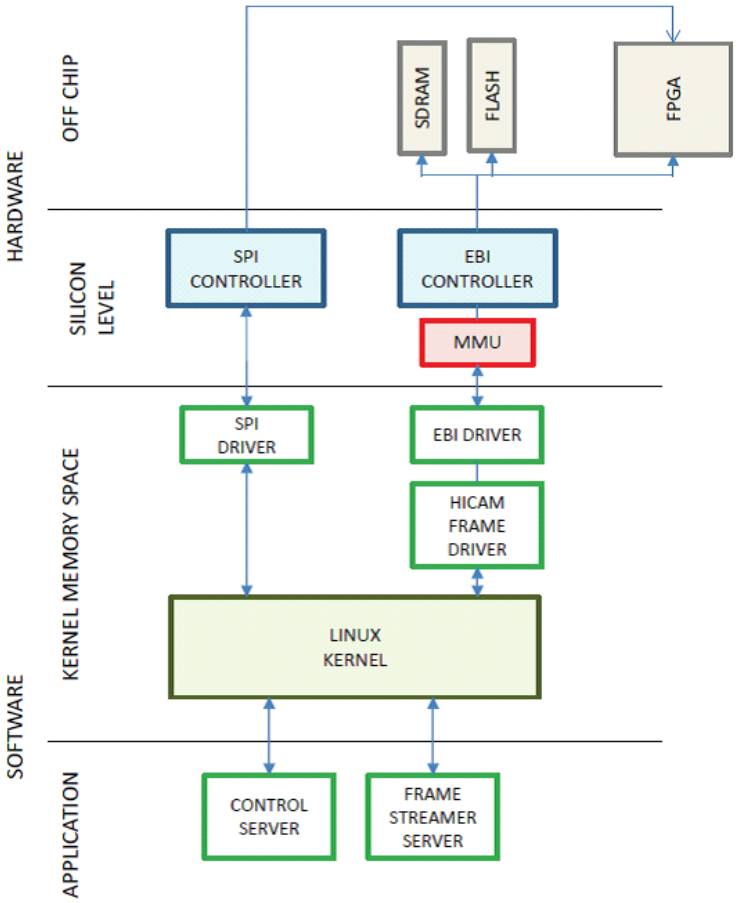


Figura 3-22 Architettura software e hardware che implementa il sistema di comunicazione tra il PC e l'FPGA utilizzando il processore AP7000

Il device driver SPI è un componente standard del sistema operativo. Esso esporta alcune funzioni di configurazione (ad esempio per impostare la velocità di bus, la posizione dell'MSB, ecc) e di scrittura e lettura di un vettore di dati.
Lo standard SPI specifica solo il livello di trasporto del protocollo di comunicazione ma non specifica un livello di applicazione. È stato scritto un driver personalizzato EBI, che mappa il dispositivo FPGA come una memoria statica corrispondente ad un indirizzo fisico specifico (0x8000 0000). Quando lo user layer, ad esempio il frame server, richiede un nuovo pacchetto di frame, il driver prende il controllo del bus EBI e accede alla BRAM sulla FPGA. Dal punto di vista dei driver per Linux, la memoria BRAM sul FPGA non è altro che una semplice memoria, il primo byte di memoria è un byte di controllo: se il byte di controllo è 0 significa che la memoria non contiene dati validi, se è a 1 il driver può leggere il contenuto della memoria. Quando una richiesta di lettura proviene dallo user layer, il driver controlla il byte di controllo: se è 0 restituisce un "non ci sono

frames valido disponibie, se è 1, legge tutta la memoria, restituisce il frame e imposta il byte di controllo a 0 per invalidare il contenuto della memoria. I frame sono scambiati in pacchetti di 100 elementi ciascuno di 81 valori. Dal lato FPGA, il sistema deve garantire che nessun valore possa essere perso durante il processo di lettura; in questo modo la coerenza dei dati è assicurata. A questo scopo vengono utilizzati due buffer di memoria in configurazione ping-pong. Quando una memoria è collegata alla scheda Ethernet, essa è dedicata alla procedura di download e il dispositivo FPGA non cambia il suo contenuto. In questo modo, la coerenza dei dati è garantita. Mentre l'AP7000 scarica i dati dall'FPGA, esso riempie una seconda memoria con i nuovi dati campionati. La velocità di download è di circa 6 Mbyte/s mentre al rate massimo di funzionamento il sistema puo generare 21 kevent/s. Ogni frame necessita di 162 byte per ospitare 81 valori. La massima velocità di scrittura è quindi di 3,4 Mbyte/s, che è inferiore a quella di lettura. Non è di conseguenza possibile riempire il buffer e si garantisce così l'integrità dei dati.
Quando l'FPGA termina la scrittura su una memoria, imposta a 1 il suo byte di controllo e inizia a leggere (in polling) la prima posizione della memoria di lettura attendendo che il driver sull'AP7000 la rilasci. Una volta che l'FPGA rileva 0 nel byte di controllo della memoria condivisa, avviene lo scambio della memoria di lettura con la memoria quella di scrittura ripristinando così il processo di comunicazione. Se nessun client è connesso al processore Atmel, il driver ferma il processo di scambio dati con l'FPGA. Il buffer dell'FPGA si riempie e i nuovi frame in arrivo dai rivelatori vengono scartati. L'architettura del buffer a ping-pong implementata nel dispositivo FPGA è mostrata in figura 3.23. I dati raccolti nell'FPGA vengono trasmessi al PC in pacchetti di 100 frames, il cui inizio è segnato da quattro byte di allineamento.

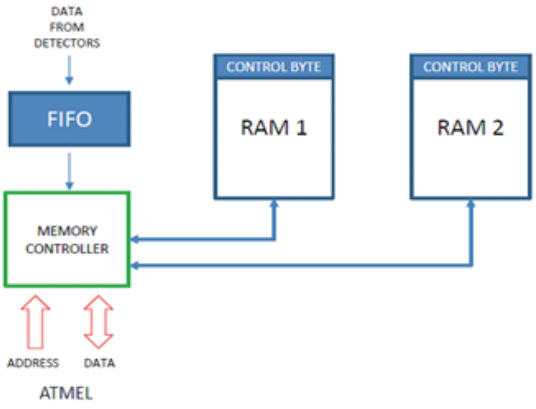


Figura 3-23 Architettura del buffer a ping-pong implementata nel dispositivo FPGA. Il buffer garantisce la coerenza e l'integrità dei dati campionati generati dai rilevatori.

Capitolo 4

ESA Control Board

Per controllare tutti i parametri configurabili dell'intero sistema, sia i registri dell'FPGA sia il registro di programmazione dei 3 ASIC, è stato costruito in Visual Basic un pannello di lavoro che allacciato al Bus SPI tramite la scheda Ethernet permette di avere un interfaccia utente.

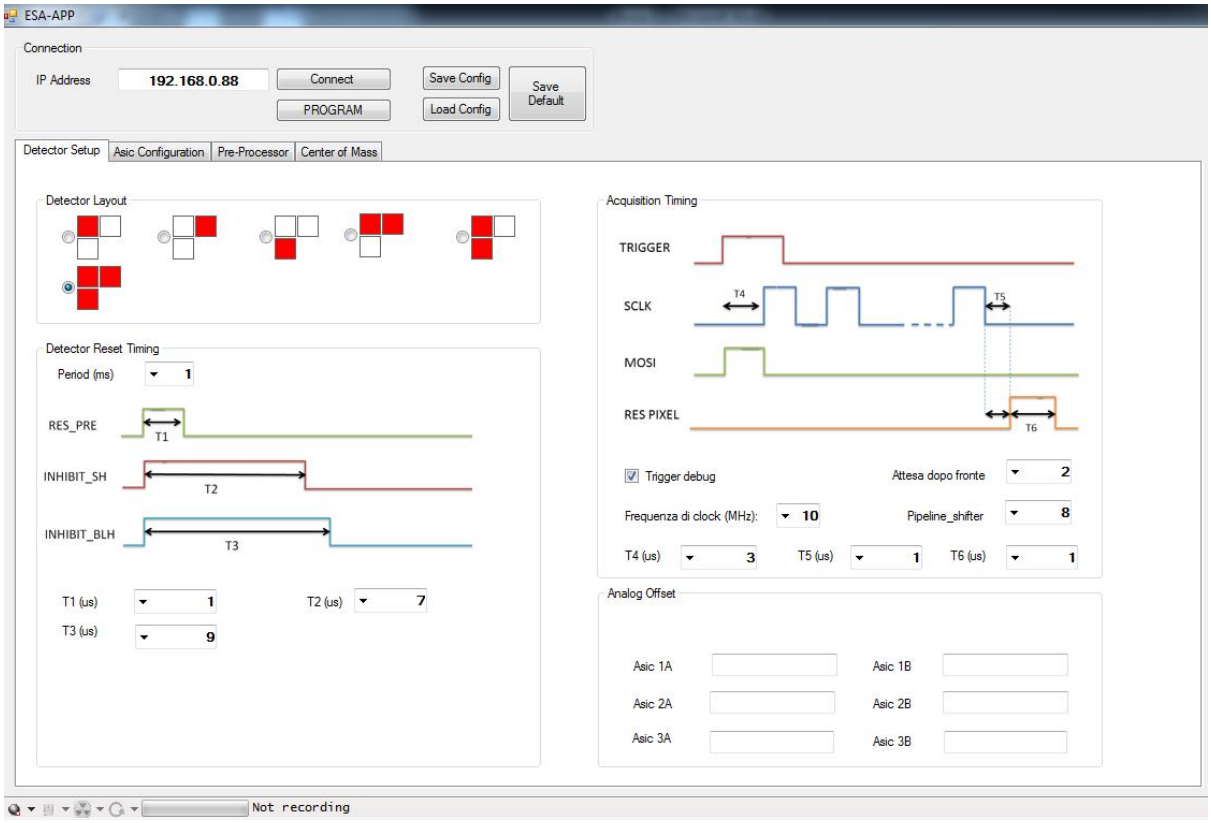


Figura 4-1 Esa Control Board (a)

Nella pagina di setup è possibile decidere tutte le temporizzazioni delle delle fasi di reset e di acquisizione, compresa la frequenza col quale acquisire e quindi campionare i detector. Anche l'offset che i DAC vanno a regolare nella scheda analogica è programmabile in questa sezione.

Nella seconda pagina del pannello sono presenti tutti i parametri di configurazione degli ASIC descritti in precedenza:

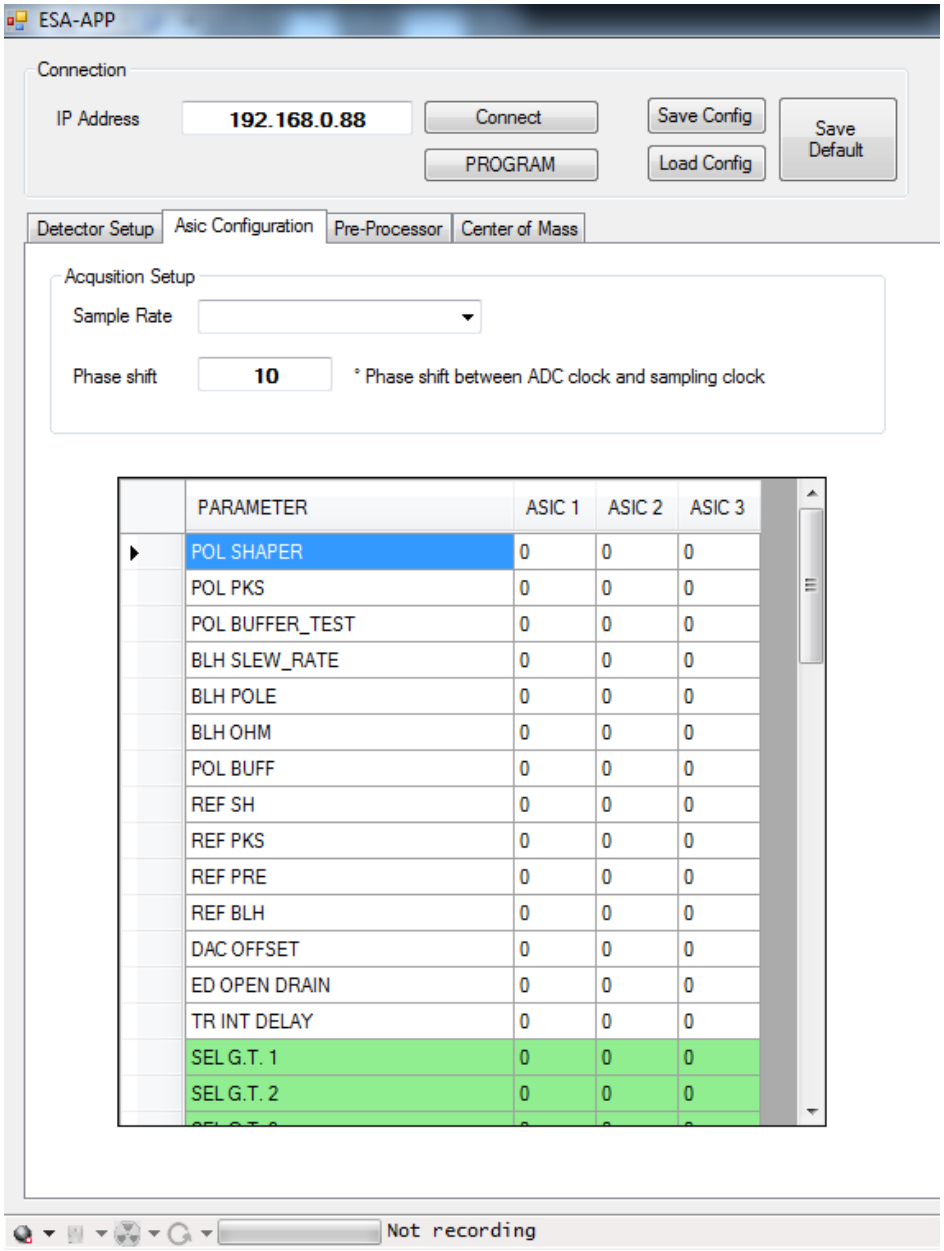


Figura 4-2 ESA Control Board (b)

I tasti di Save Config e Load Config permettono di non perdere le configurazioni inserite e di riconfigurare facilmente il sistema durante il lavoro di debug.
Un ulteriore applicazione molto semplice si occupa di dare lo start all'acquisizione e di inserire i dati in arrivo in un file di testo leggibile con Matlab.

Capitolo 5

Descrizione di un FPGA

Aspetti introduttivi di un FPGA

Negli ultimi tre decenni i circuiti integrati hanno rispettato il tasso di crescita empiricamente descritto dalla legge di Moore. Tale legge, che prende il nome dal suo formulatore Gordon E. Moore, uno dei fondatori di Intel, prevede che ogni diciotto mesi circa raddoppi il numero massimo di transistori integrabili nei circuiti integrati senza un corrispondente aumento dei costi dei chip. Questo raddoppio periodico di densità si traduce, tra l'altro, in un aumento delle prestazioni a parità di costo. Per esempio, nel caso dei microprocessori si verifica che al raddoppio di densità di transistori corrisponde un incremento delle prestazioni del 100% circa. L'evoluzione esponenziale delle prestazioni osservata negli anni ha condotto ad un livello tale che, già oggi e sempre più nel prossimo futuro, il rispetto della legge di Moore non è il solo obiettivo primario nello sviluppo dei circuiti integrati. Oggi le più avanzate conoscenze tecnologiche rappresentano un bagaglio comune ai costruttori di circuiti integrati e ciò comporta che non siano più le semplici prestazioni in termini di numero di porte integrate a decretare il successo commerciale di un prodotto, come pure la reperibilità di potenti strumenti di progettazione CAD non rappresenta più un ostacolo sostanziale. Importante, e in molte applicazioni fondamentale, è divenuta la capacità di realizzare prodotti in grado di essere sviluppati e adattati alle esigenze del mercato con la rapidità con cui si evolvono praticamente tutti i settori dell'elettronica, dal consumer alle applicazioni specialistiche. In questo scenario si rivela dunque decisivo per i costruttori di circuiti integrati perseguire, come primari, due obiettivi: customization e time-to-market. La proprietà di customization è la potenzialità di un prodotto di essere adattato a diverse esigenze applicative con il minimo intervento sulla struttura hardware dello stesso. La proprietà di time-to-market è la capacità di ridurre al minimo il tempo che intercorre tra la domanda di un prodotto e la sua disponibilità sul mercato. Le richieste su entrambi i fronti diventano sempre più esigenti e condizionanti l'attività di progettazione e di produzione. Nell'ambito dell'elettronica digitale, i

dispositivi logici programmabili sono, insieme ai microprocessori, i circuiti integrati che meglio si prestano a soddisfare tali esigenze del mercato. Infatti la programmabilità dei dispositivi logici programmabili è alla base della filosofia della progettazione di tali dispositivi, ovvero si tende a sviluppare rapidamente i sistemi su basi semplici che, successivamente, vengono specializzati per incontrare le esigenze funzionali specifiche. Si stima che oggi oltre il 70% dei nuovi sistemi vengano progettati con la presenza di almeno un dispositivo logico programmabile a bordo. I dispositivi logici programmabili principali in termini di prestazioni e capacità sono le FPGA (Field Programmable Gate Array), che oggi sono a tutti gli effetti elementi primari e spesso insostituibili dei sistemi elettronici digitali. Impiegati inizialmente per integrare logica sparsa di servizio nei sistemi, le FPGA sono divenuti oggi elementi di primaria importanza nel progetto di sistemi digitali e, tra i dispositivi logici programmabili, sono quelli che più efficacemente si prestano, insieme ai DSP, a soddisfare le tendenze del mercato verso customization e time-to-market. La tecnologia delle FPGA è cresciuta in questi ultimi anni a tal punto che in molte applicazioni esse hanno avuto il sopravvento sugli ASIC (Application Specific Integrated Circuit) Gate Array. I vantaggi nell'utilizzo delle FPGA sono evidenziati dalla mancanza dei costi di mascheratura, dal drastico abbattimento dei tempi di attesa per i prototipi, dalla riduzione dei rischi connessi con la tecnologia non riprogrammabile e dalla versatilità offerta dalla riprogrammabilità, sia nell'evoluzione del progetto sia durante l'uso del sistema. La linea di frontiera che limita la convenienza in termini economici delle FPGA rispetto agli ASIC Gate Array è principalmente fissata da due fattori: da un lato il numero di pezzi prodotti e dall'altro la densità di porte logiche necessarie per implementare il progetto. Quanto più questi parametri crescono tanto più rendono conveniente l'impiego degli ASIC Gate Array. In termini tecnologici, infatti, gli ASIC Gate Array ottimizzano l'uso delle risorse di silicio mentre le FPGA mettono in conto all'utente anche le parti non impiegate sia di logica sia di interconnessione.

Attualmente almeno tre società al mondo si sono specializzate nella produzione di questi dispositivi il cui uso sta continuamente crescendo: Xilinx, Altera e Lattice. In questo progetto è stata usata una FPGA della Xilinx e, dunque, nel seguito saranno descritte solo le FPGA prodotte da questa azienda. Tuttavia il concetto che sta alla base di questi dispositivi è comune: tutti questi circuiti integrati sono organizzati internamente in una matrice di piccoli blocchi circuitali elementari programmabili e da una rete di connessioni anch'esse

programmabili che li connettono tra di loro e con il mondo esterno, ovvero con i piedini d'uscita del dispositivo. Lo scopo di una simile organizzazione è evidente: si vuole fornire all'utente una piattaforma hardware integrata, dunque estremamente veloce e capiente, totalmente configurabile su cui si possa implementare qualunque architettura hardware digitale che faccia parte del sistema in progetto. Pertanto l'uso di dispositivi FPGA garantisce al progettista di avere integrati general-pourpose, cioè non specializzati per una particolare applicazione. Questo massimizza il range di applicazioni possibili e quindi il mercato disponibile per questi prodotti, con conseguente abbattimento dei prezzi. La specializzazione dell'applicazione è affidata all'utente; questo offre sia la possibilità di differenziare i propri prodotti utilizzando di fatto la stessa piattaforma hardware, sia la possibilità di fare evolvere il prodotto semplicemente implementando circuiti digitali sempre più evoluti nello stesso dispositivo, ovviamente nei limiti di quest'ultimo. Le moderne FPGA offrono di fatto la possibilità di realizzare propri ASIC e di poterli modificare a proprio piacimento. Inoltre la loro dimensione è cresciuta a tal punto da consentire in molti casi l'integrazione di tutto il sistema digitale su uno solo di questi dispositivi, con l'unico limite imposto dalle potenze dissipate. Questo consente di aggiornare e modificare il progetto molto velocemente e di personalizzare le sue funzionalità realizzando i cosiddetti System On Chip. Per le loro caratteristiche tecniche, le FPGA sono l'hardware ottimale per qualunque progetto digitale che comporti un'elaborazione parallela pesante dell'informazione, come per esempio l'elaborazione delle immagini. All'interno del dispositivo si possono infatti configurare diversi sottocircuiti digitali operanti in parallelo e controllati da un sistema supervisore anch'esso implementato nello stesso chip.

Una prima netta classificazione di questi dispositivi può essere fatta analizzando una delle principali proprietà delle FPGA, ovvero la riconfigurabilità. Esistono infatti FPGA programmabili una sola volta OTP (One Time Programmable) e FPGA riprogrammabili. Uno dei principali vantaggi nell'uso di una struttura di programmazione OTP risiede nell'elevata tolleranza all'esposizione a radiazioni ionizzanti senza compromissione della funzionalità. Per questo le FPGA OTP sono per lo più impiegate in campo aerospaziale e militare. D'altro canto, il prezzo pagato dall'utente che ha una sola possibilità irreversibile di configurare il dispositivo è alto. In questi dispositivi, gli elementi bistabili da programmare per stabilire la configurazione sono di tipo antifuse. Gli antifuse sono, in breve, dei fusibili integrati il cui stato di conduzione viene modificato mediante il passaggio di

una corrente di opportuna intensità. Dal nome si capisce come, questi, a differenza dei normali fusibili, siano normalmente aperti e stabiliscano un collegamento tra i terminali soltanto dopo il passaggio di corrente. Oltre alla minor flessibilità in fase di progetto del prototipo, questi dispositivi risultano inutilizzabili per tutte quelle applicazioni in cui si richiede che il dispositivo possa essere riprogrammato con semplicità una volta inserito nel sistema (In-System Programming, ISP). Nel caso delle FPGA riprogrammabili, gli elementi bistabili di configurazione sono fondamentalmente di due tipi: SRAM o EEPROM. Pur offrendo entrambi la possibilità di modificare nel tempo la configurazione del dispositivo (il tipo SRAM teoricamente un numero illimitato di volte), le due tipologie si differenziano fondamentalmente in quanto il tipo SRAM non mantiene la configurazione al momento dello spegnimento del dispositivo, il quale di conseguenza deve essere configurato ad ogni accensione. Il tipo EEPROM invece, consente una configurazione non volatile del dispositivo, stabile anche in assenza di alimentazione. Purtroppo, entrambe le soluzioni, si dimostrano particolarmente vulnerabili agli effetti dell'esposizione a radiazioni ionizzanti, che possono determinare l'alterazione dell'informazione immagazzinata. Ovviamente le FPGA hanno anche degli svantaggi operativi che possono essere riassunti nei due seguenti aspetti. Per prima cosa l'estrema riconfigurabilità (praticamente l'FPGA si può riprogrammare completamente, fino alla singola connessione tra un blocco logico e l'altro) comporta la presenza di connessioni sul chip con moltissimi transistori di diramazione. Le connessioni programmabili sono infatti realizzate con matrici di connessioni in metal alle cui intersezioni sono presenti dei pass-transistor. Questi, a seconda dello stato che assumono, modificano il percorso del segnale. Il grande numero di transistori aumenta la resistenza equivalente della connessione e, dunque, limita la velocità di propagazione del segnale digitale su di essi. Tale problema non si presenta nel caso in cui si utilizzino dei dispositivi ASIC i quali, dispongono di connessioni in puro metal che, progettate per ottimizzare la specifica funzione da svolgere, offrono una resistenza parassita estremamente bassa. Questa caratteristica permette di ottenere velocità di funzionamento particolarmente elevate se paragonate a quelle delle FPGA. Tuttavia le FPGA hanno ampie possibilità di sopperire a questo limite tramite l'esecuzione in parallelo delle operazioni, ad esempio. L'altro aspetto consiste nel fatto che il processo di generazione del progetto su FPGA si basa su sistemi di sintesi circuitale altamente automatizzati che necessitano di computer molto potenti per poter funzionare in maniera efficiente e

rapida. Inoltre, il progetto deve essere totalmente simulato per verificare il suo comportamento e la sua robustezza. La simulazione è, da un punto di vista computazionale, molto complessa e, anche per essa, sono necessari computer molto potenti. Ovviamente ciò è proporzionato alla complessità dell'architettura da implementare nel dispositivo.

Architettura della Spartan3A

Il dispositivo FPGA utilizzato nel progetto appartiene alla famiglia Spartan3A. Le specifiche dei vari dispositivi appartenenti alla famiglia sono riassunte nella seguente tabella.

Device	System Gates	Equivalent Logic Cells	CLBs	Slices	Distributed RAM bits	Block RAM bits	In-System Flash bits	Dedicated Multipliers	DSP48As	DCMs	Maximum User I/O
XC3S50A/AN	50K	1,584	176	704	11K	54K	1M	3	-	2	144
XC3S200A/AN	200K	4,032	448	1,792	28K	288K	4M	16	-	4	248
XC3S400A/AN	400K	8,064	896	3,584	56K	360K	4M	20	-	4	311
XC3S700A/AN	700K	13,248	1,472	5,888	92K	360K	8M	20	-	8	372
XC3S1400A/AN	1400K	25,344	2,816	11,264	176K	576K	16M	32	-	8	502
XC3SD1800A	1800K	37,440	4,160	16,440	260K	1,512K	-	-	84	8	519
XC3SD3400A	3400K	53,712	5,968	23,872	373K	2,268K	-	-	126	8	469

Tabella 5-1 Elenco dei dispositivi della famiglia Spartan3A con relative caratteristiche.

L'architettura delle Spartan3A è composta dai seguenti elementi funzionali: i CLB (Configurable Logic Block) con i quali vengono implementate le funzioni logiche, i IOB (Input Output Block) responsabili del controllo del flusso dei dati nel dispositivo, il DCM (Digital Clock Manager) il quale gestisce il segnale di clock all'interno del dispositivo, la Block RAM, utilizzata per la memorizzazione dei dati e il Multiplier o DSP48A per eseguire moltiplicazioni.

La logica riconfigurabile

I blocchi logici configurabili sono la maggior risorsa logica utilizzabile per l'implementazione di circuiti sequenziali e combinatori e compongono l'intera struttura interna della FPGA secondo uno schema a matrice.

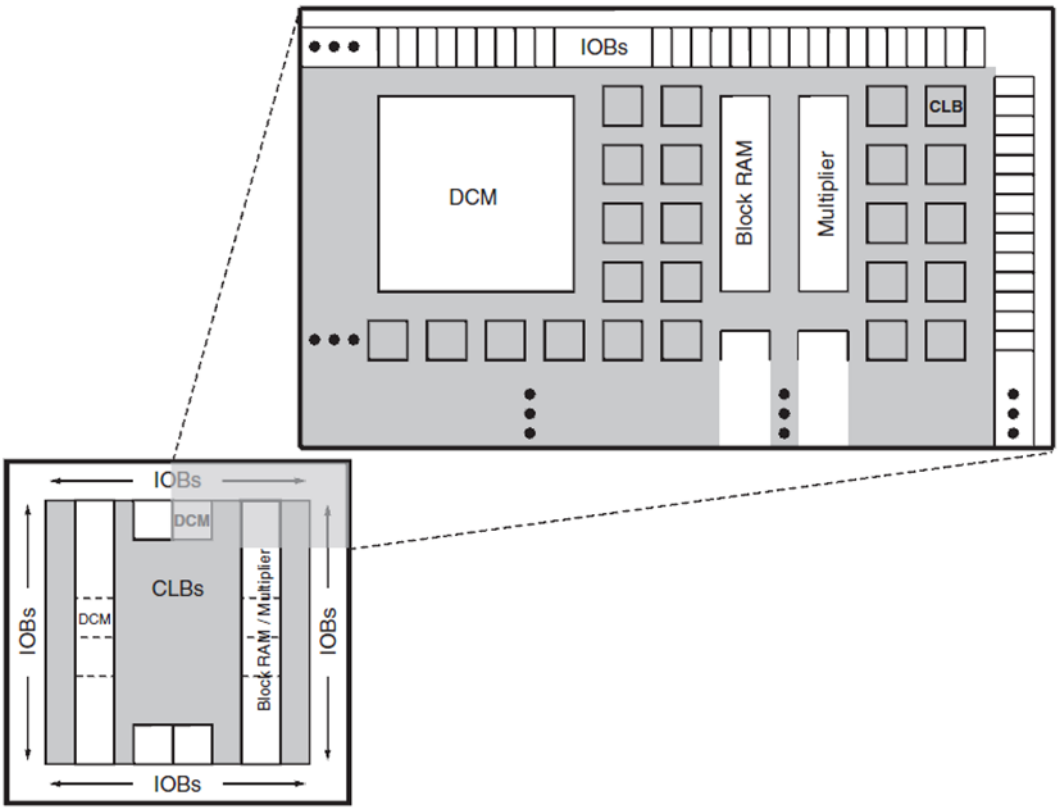


Figura 5-1 Illustrazione della struttura a matrice delle FPGA

Ogni CLB comprende quattro slices, raggruppate a coppie, ed è connesso alla matrice di switch per l'accesso alla rete di connessione globale. Le coppie sono organizzate in colonne e, pur appartenendo allo stesso CLB, non dispongono di nessun collegamento diretto fra di loro. L'unica connessione diretta presente fra slices adiacenti è la linea di riporto di ingresso e uscita. All'interno del blocco le slices vengono indicate con la seguente modalità. Una 'X' seguita da un numero indica la colonna, in ordine crescente da sinistra a destra, mentre una 'Y' seguita da un numero indica la posizione all'interno della coppia. Partendo dal basso, vengono usati 0,1 per la prima riga, 2 e 3 per la seconda e così via. Per ogni CLB, la coppia situata sulla sinistra viene chiamata SLICEM mentre quella sulla destra SLICEL.

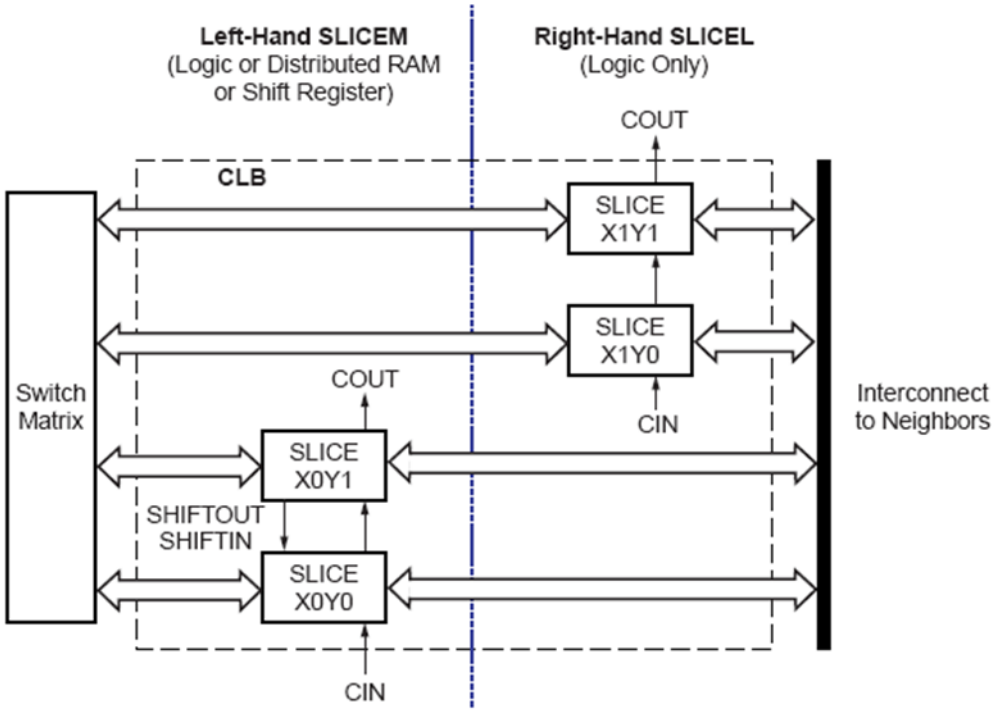


Figura 5-2 Diagramma a blocchi di un CLB

All'interno di ogni slice sono presenti due generatori di funzioni logiche (LUT) accompagnati da altrettanti elementi di memoria, diversi multiplexer e la logica di riporto. Questi componenti vengono utilizzati per implementare funzioni logiche, aritmetiche e blocchi di memoria ROM. In aggiunta ogni SLICEM è in grado di immagazzinare dati attraverso una memoria RAM e implementare uno shift register. Le funzioni che possono essere implementate nelle slices sono ottenute per mezzo LUT (Look-Up Table). Le Spartan3A presentano quattro ingressi e una sola uscita, dando così la possibilità di eseguire qualsiasi operazione logica avente quattro ingressi. Oltretutto è possibile combinare più LUT, grazie alla presenza di numerosi multiplexer, per eseguire funzioni aventi più operandi.

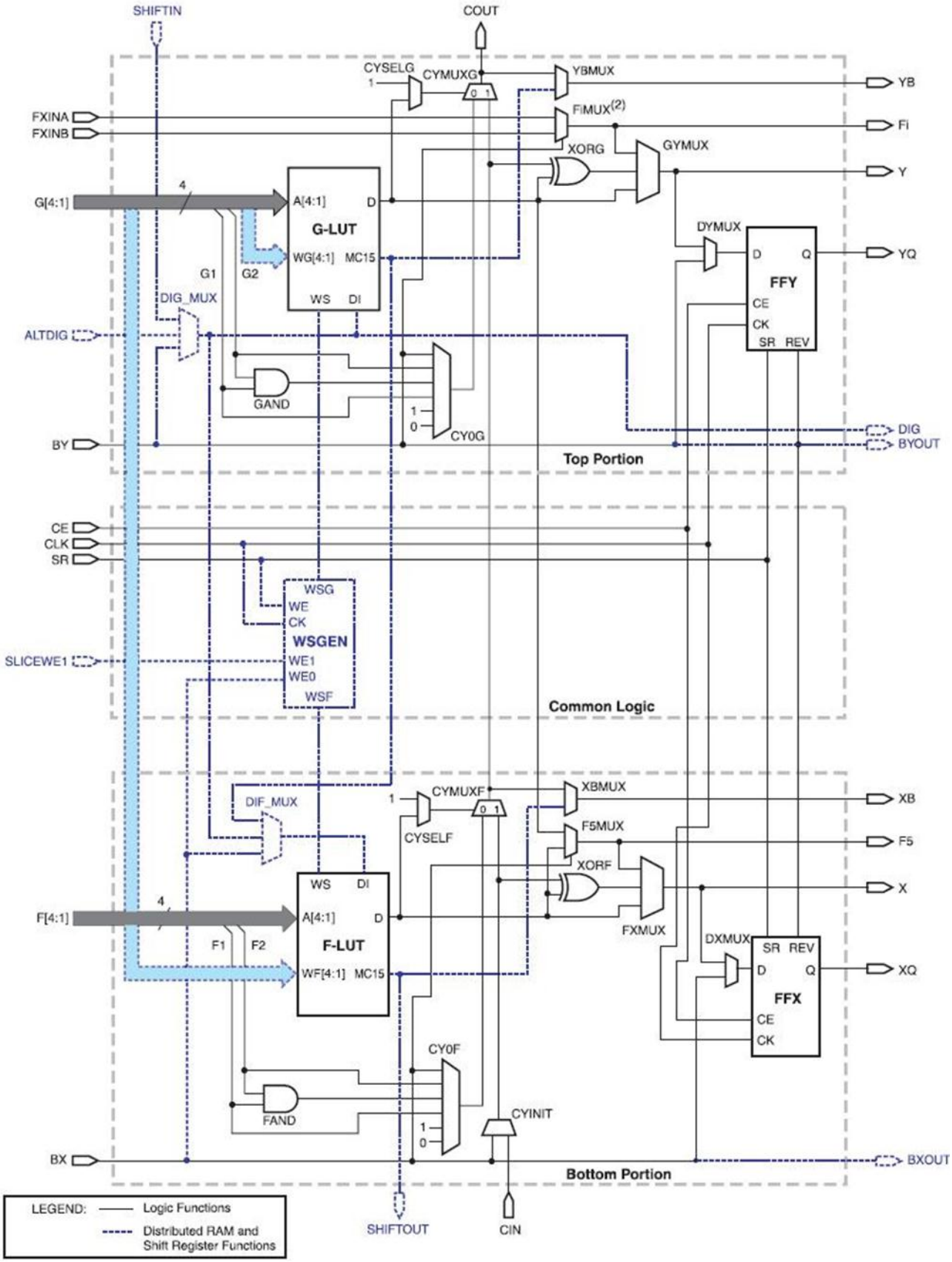


Figura 5-3 Struttura interna dettagliata di una SLICEM.

Ogni slice appartenente alla coppia sinistra può essere configurata per comportarsi come una vera e propria memoria RAM distribuita. Parlando di memoria distribuita ci si riferisce alla memoria utilizzabile all'interno delle FPGA implementata sfruttando le LUT nelle SLICEM. Essa è rapida, localizzata e ideale per la gestione di piccole quantità di dati. Infatti con una LUT è possibile immagazzinare sedici bit e combinando più unità è possibile ampliare la dimensione. La memoria distribuita scrive sincronamente ma legge asincronamente; in caso di necessità comunque è possibile implementare una funzione per ottenere la lettura sincrona. Combinando due LUT è possibile creare una RAM con due porte, una di lettura e scrittura e l'altra di sola lettura.

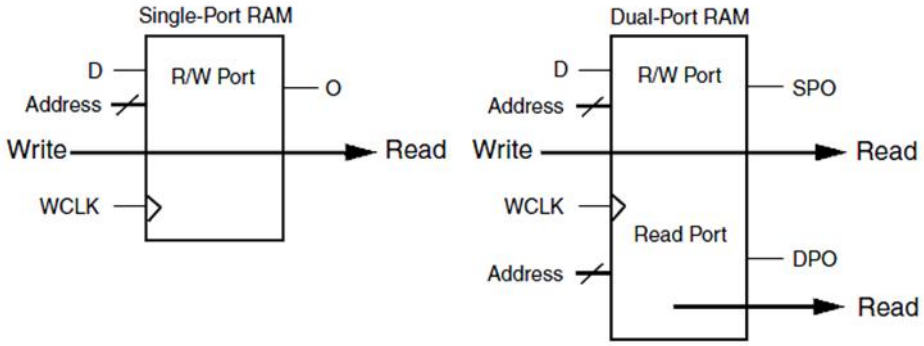


Figura 5-4 RAM distribuita a singola porta e a doppia porta

Come già accennato in precedenza, è possibile programmare ogni LUT dello SLICEM come uno shift register a sedici bit. In questa maniera è possibile ritardare un dato seriale da uno a sedici cicli di clock; combinando più LUT è possibile aumentare l'entità del ritardo. Indipendentemente dal tipo (SLICEM o SLICEL) inoltre, ogni slice può implementare, sempre grazie alle LUT, un banco di memoria ROM. Con il termine ROM utilizzato in questo contesto, si intende una memoria di sola lettura dal lato sistema ma che viene inizializzata e riscritta ogni volta che viene caricato il codice all'avvio. Non è quindi quella che normalmente viene indicata come ROM nei comuni sistemi elettronici in cui la scrittura avviene un'unica volta.

Block RAM

In progetti che richiedono una gran quantità di memoria risulta improponibile utilizzare come memoria solo la RAM distribuita in quanto ciò comporterebbe uno spreco eccessivo di blocchi altrimenti sfruttabili per implementare della logica. A tal scopo esistono dei blocchi dedicati in grado di memorizzare 18 Kbit ognuno, per la

famiglia Spartan3A. Tutti questi blocchi, il cui numero dipende dal dispositivo impiegato, sono disposti in colonna sul chip in modo da permettere una connessione gli uni con gli altri molto semplice e veloce. In questo modo è possibile ottenere una maggiore profondità e ampiezza di memoria con una penalizzazione in termini di velocità minima. È possibile implementare una memoria a doppia o a singola porta; a seconda delle esigenze inoltre è possibile impostare due clock differenti e due dimensioni dei dati differenti in modo da consegnare al progettista una grande flessibilità. Anche il protocollo di scrittura in memoria lascia ampio spazio di manovra all'utente. Esistono infatti tre differenti tipologie di scrittura che si differenziano a seconda del dato che viene presentato sulla porta di uscita a seguito di un'operazione di write:

- write first : sulla porta di lettura viene fornito il dato appena scritto;
- read first : sulla porta di lettura viene posto il dato precedentemente memorizzato (che quindi `e stato sovrascritto);
- no change: in uscita viene mantenuto il dato letto al ciclo precedente.

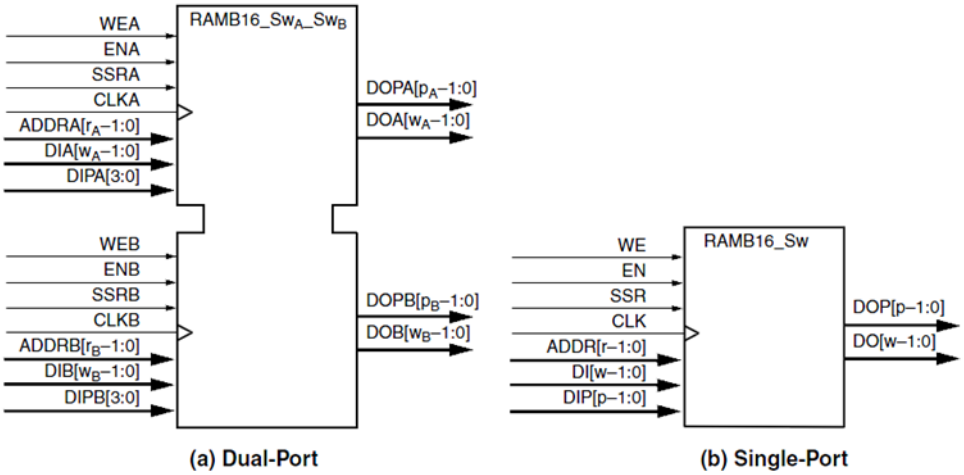


Figura 5-5 Block RAM

Interfaccia esterna

Qualsiasi sia il sistema implementato all'interno della FPGA, esso deve interfacciarsi col mondo esterno e scambiare con esso delle informazioni. Nelle FPGA nessun componente interno viene messo in diretta connessione con i terminali esterni del dispositivo per una serie di motivi:

- è noto che trasportare segnali digitali dall'interno all'esterno di un circuito integrato presuppone l'introduzione di architetture a cascata di buffer, che assicurino una adeguata capacità di corrente dinamica verso l'esterno senza sovraccaricare capacitivamente il circuito verso l'interno;
- per garantire una ampia applicabilità del prodotto si deve poter scegliere lo standard logico in uscita più adatto al progetto.

Questi vincoli presuppongono l'esistenza a bordo della FPGA di dispositivi di interfacciamento con l'esterno denominati IOB (Input Output Block). Questi blocchi garantiscono un'interfaccia programmabile e bidirezionale tra i pin e la logica interna. Ogni IOB presenta tre percorsi per il segnale: input, output e 3-state; ognuno di essi è munito di elementi di memoria che possono agire da registri o latch. Questi blocchi vengono disposti all'interno della FPGA secondo una struttura ben definita e ordinata: infatti ricoprono il perimetro della matrice interna di CLB. Oltre alla gestione di connessioni standard è possibile configurare ingressi ed uscite differenziali ad alta velocità. Questa caratteristica è ottenuta dalla collaborazione di due IOB. Una peculiarità aggiuntiva di questi blocchi è l'implementazione della tecnologia DCI (Digital Controlled Impedance). Questa tecnologia permette appunto di controllare digitalmente l'impedenza delle terminazioni in modo che si adatti al meglio all'impedenza della linea di trasmissione onde evitare fenomeni di riflessione indesiderati.

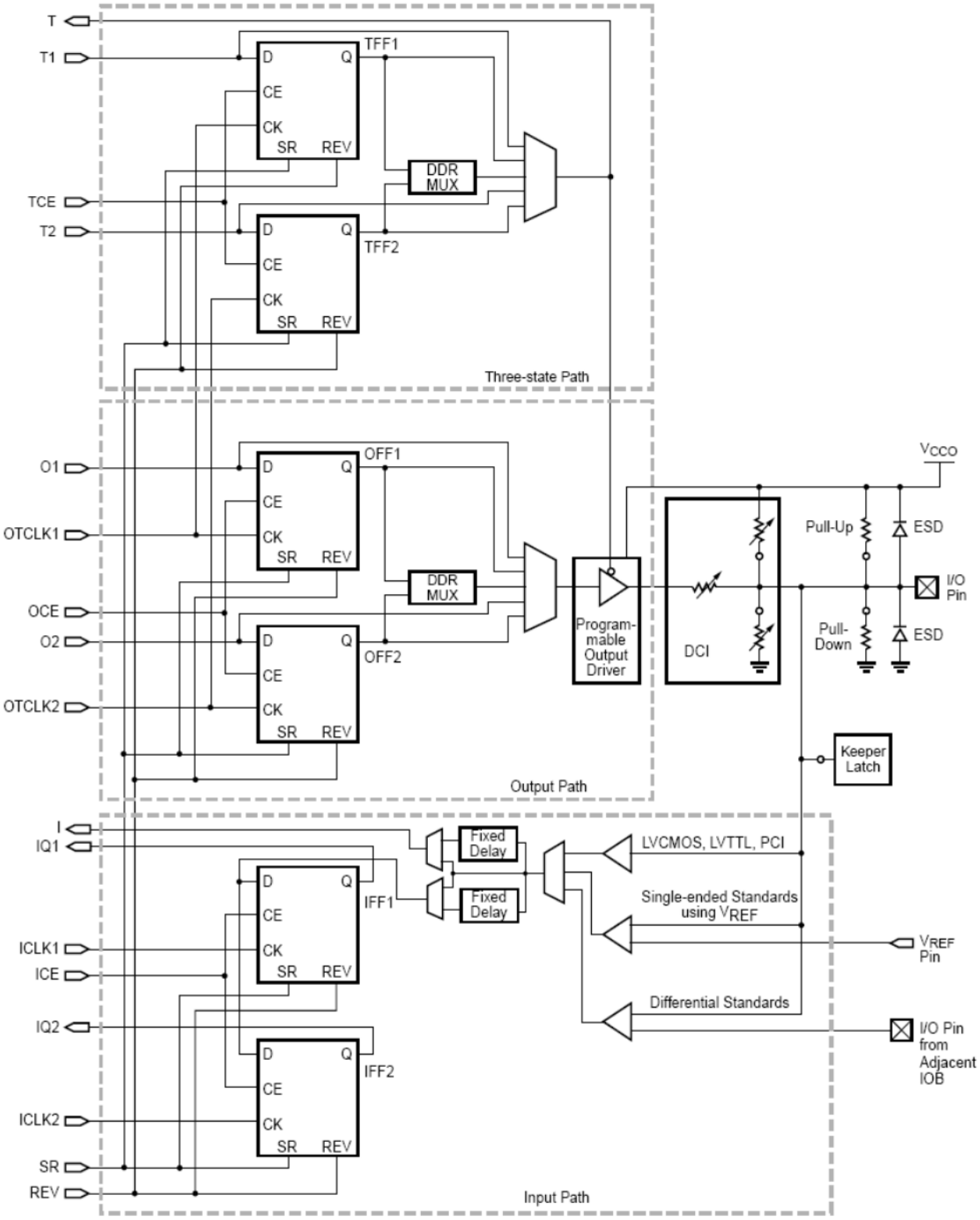


Figura 5-6 Diagramma di un blocco IOB

I gestori del clock

I segnali di clock sono i segnali più importanti in un sistema digitale. Essi trasportano l'informazione temporale di sincronizzazione tra i vari dispositivi in maniera che essi possano operare correttamente. E' infatti naturale pensare a un processo di elaborazione digitale dell'informazione come un processo diviso in fasi temporali successive e che evolve nel tempo. Le FPGA possono lavorare con più segnali di clock differenti contemporaneamente ed è quindi necessario prestare molta cura nella gestione di tali segnali. Le proprietà fondamentali richieste ai segnali di clock in un sistema digitale sono le seguenti:

- periodo e duty cycle (percentuale di tempo rispetto al periodo in cui il segnale è alto) estremamente precisi;
- basso jitter (scostamento del duty cycle tra un ciclo e il successivo), cioè il segnale deve presentare delle differenze di duty cycle tra un periodo e il successivo estremamente ridotte;
- basso skew (differenza di tempo di arrivo del fronte di clock tra due punti della rete di distribuzione del clock), cioè il segnale di clock deve arrivare in tutte le parti del dispositivo nello stesso istante.

Ogni dispositivo della famiglia Spartan3 dispone di 8 generatori clock globali con altrettante linee di distribuzione le quali possono sincronizzare tutte le risorse nel dispositivo (CLB, BRAM, IOB...). Tali linee sono progettate specificatamente per collegare tutti gli ingressi di clock con i vari blocchi interni prestando particolare attenzione a fattori quali lo skew e il jitter. Inoltre sono presenti otto multiplexer globali i quali selezionano segnali provenienti dal Global Clock o dal DCM (Digital Clock Manager). I blocchi DCM sono pensati anch'essi come veri e propri CLB e, tra le numerose funzioni che possono fornire, le principali sono:

- Clock Deskew. Grazie ad un DLL (Delay Locked Loop) integrato è possibile eliminare completamente i ritardi di distribuzione del clock. Il DLL infatti contiene elementi di ritardo (semplici buffer) e logica di controllo. Il segnale di clock d'ingresso pilota la catena di elementi di ritardo in modo che all'uscita di ogni elemento sia presente il segnale d'ingresso ritardato di un certo periodo. Per mezzo di un rivelatore di fase si confronta il segnale d'ingresso con un opportuno segnale di feedback e viene selezionata l'uscita della catena più opportuna affinché lo sfasamento dei due sia nullo.
- Frequency Synthesis e Phase Shifting. Esistono per prima cosa due uscite ausiliarie in grado di fornire un segnale ad una

frequenza doppia (sia in fase che in controfase) di quella di ingresso. Una ulteriore uscita fornisce un segnale ad una frequenza che corrisponde ad una specifica frazione di quella di ingresso. Infine altre due uscite possono fornire una frequenza (sempre in fase o controfase) ricavata da moltiplicazioni e divisioni multiple della fondamentale.

- Dinamic reconfiguration. Grazie ad uno speciale BUS che collega tutti i DCM è possibile riconfigurare ogni singolo DCM senza modificare il resto del sistema.

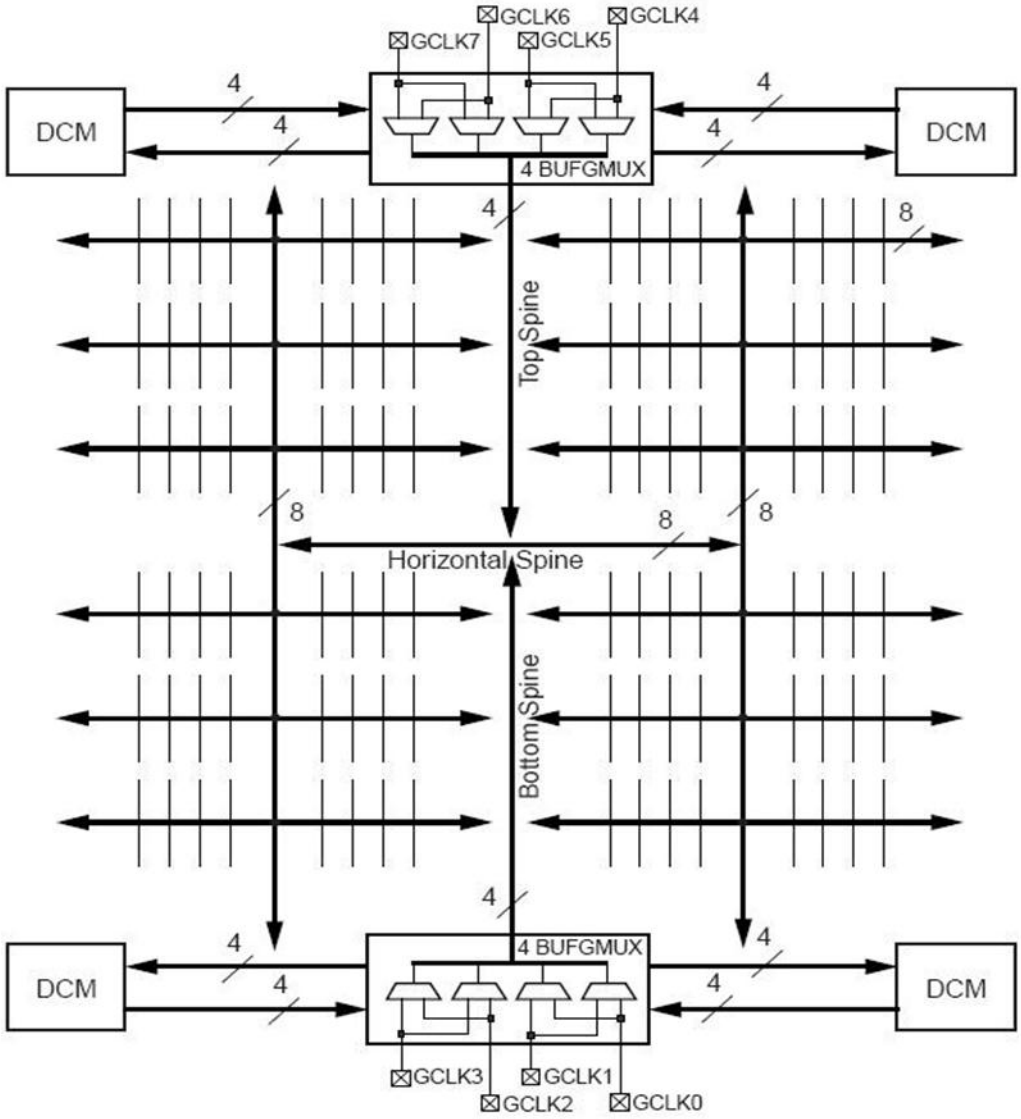


Figura 5-7 Distribuzione del segnale di clock all'interno della Spartan3.

Le interconnessioni programmabili

Tutti i CLB e gli IOB devono poter essere elettricamente connessi tra di loro per poter scambiare dati e, quindi, costituire il sistema digitale. A bordo delle FPGA, tutte le connessioni devono essere programmabili e consentire a un qualunque blocco di raggiungerne un qualsiasi altro presente sul dispositivo. Affinché quanto illustrato venga realizzato è necessario che ciascun CLB e ciascun IOB abbia associata una matrice di connessione denominata GRM (General Routing Matrix) situata in prossimità ad esso. Questa può essere vista, di fatto, come un ulteriore blocco che raccoglie tutte le connessioni del CLB o dell'IOB associato e lo collega con il resto del sistema. Le GRM si trovano ovviamente sui nodi della rete di interconnessione globale. Uno dei punti deboli delle FPGA è sempre stato il ritardo di propagazione introdotto dalla rete configurabile di interconnessioni dovuto al gran numero di punti di diramazione e, quindi, di resistenze equivalenti introdotte sui connettori. Si può considerare che tanto più è elevato il numero di matrici di interconnessione attraversate da una risorsa di connessione, tanto più questa introdurrà ritardo nella propagazione del segnale. Proprio per sopperire a questo problema, anche in questo caso, come per la gestione dei segnali di clock, si utilizzano strutture molto complesse per la loro ottimizzazione. Vengono impiegati generalmente quattro tipi di connessioni differenti adatte ognuna ad un particolare tipo di collegamento:

- Long lines, che connettono tra di loro le matrici di connessione di 6 in 6;
- Hex lines, che connettono tra di loro le matrici di 3 in 3;
- Double lines, che connettono tra di loro matrici di 2 in 2;
- Direct lines, che connettono tra di loro direttamente ogni CLB con i vicini.

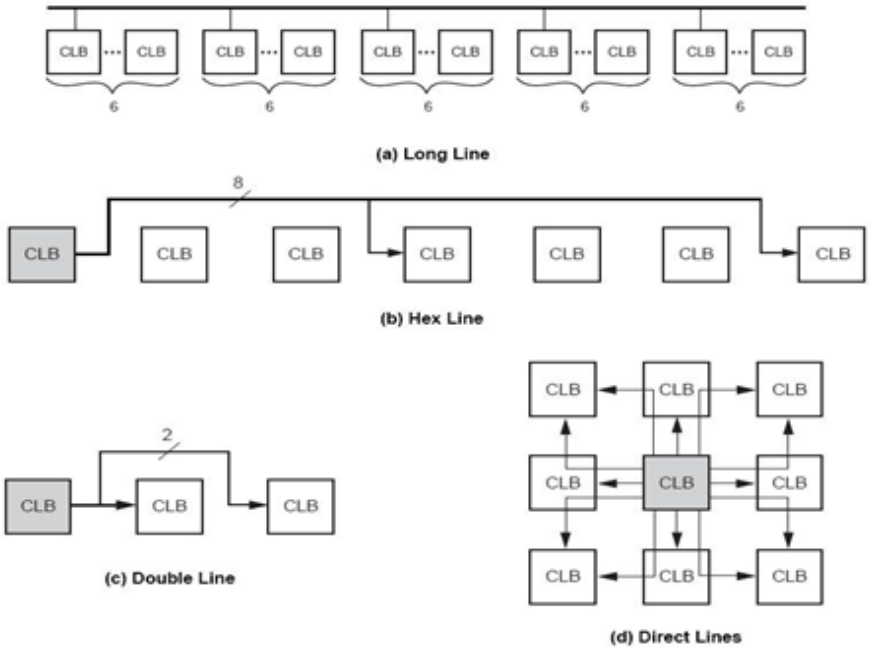


Figura 5-8 Schema dei quattro tipi di interconnessione.

La programmazione dell'Hardware

L'intero insieme di fasi di configurazione di un dispositivo logico programmabile, dal progetto alla programmazione, è completamente computer assisted. Ciò non solo costituisce un enorme vantaggio, ma sarebbe anche impossibile fare altrimenti vista la complessità strutturale congenita di progetti anche di limitata difficoltà. I software tools necessari sono disponibili sia modulari, ciascuno specializzato per ogni fase del processo, sia in forma integrata. Concettualmente tutti effettuano le medesime operazioni. L'iter di configurazione è descritto nel diagramma di flusso riportato nella figura sottostante.

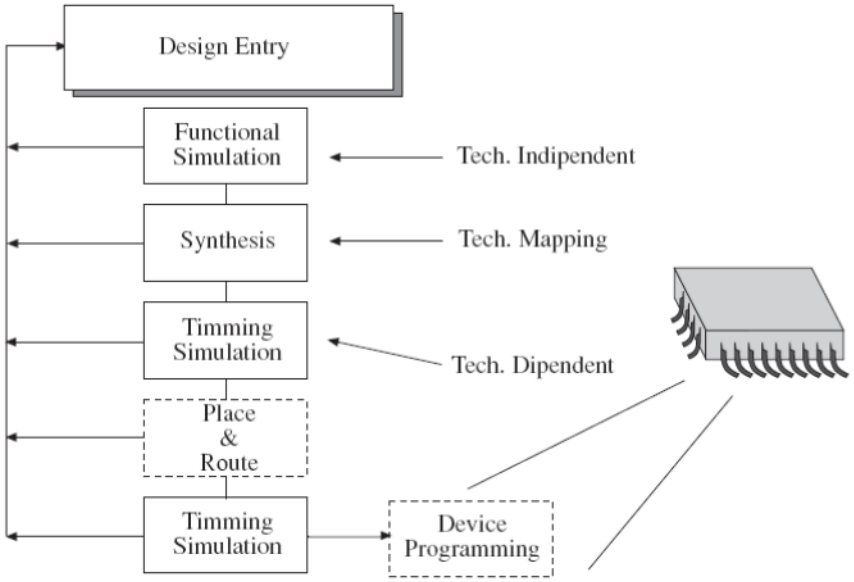


Figura 5-9 Flusso di programmazione di un FPGA

Design entry

La prima fase è definita design entry e ha la finalità di comunicare al sistema di programmazione il progetto da implementare nel dispositivo. Tale comunicazione consiste nella trasposizione in formato elettronico della rete logica corrispondente al progetto. Esistono tre modalità canoniche per descrivere il progetto in formato elettronico: mediante schematico, attraverso linguaggi descrittivi di alto livello e mediante diagrammi di stato. La descrizione mediante schematico consiste nel disegnare sul foglio elettronico dell'editor del software specifico lo schematico a 'porte logiche' del sistema da implementare, dove al termine 'porte logiche' si attribuisce tanto l'accezione di porte logiche elementari quanto quella di complessi blocchi logici strutturati di libreria. L'uso di linguaggi descrittivi è senza dubbio la via più efficiente, e quindi la più usata dagli utenti, per introdurre il progetto. La descrizione di un progetto attraverso le istruzioni di un codice standard ad alto livello fornisce tutti i vantaggi dell'uso di un linguaggio di programmazione strutturato. In particolare, nel caso dei dispositivi logici programmabili il progetto risulta in tal modo adatto a essere facilmente modificato, esportato, gerarchizzato e ne è accelerata la fase di sviluppo, soprattutto nel caso di grandi progetti in cui al team di sviluppo viene garantita un'interfaccia standard tra i diversi progettisti. Inoltre, la descrizione del progetto mediante un linguaggio è completamente indipendente dal dispositivo elettronico prescelto (technology independent), in quanto non viene considerato alcun componente reale ma ne viene emulato il solo aspetto logico funzionale. Di conseguenza anche la simulazione del progetto descritto risulta totalmente technology

independent, fattore questo che garantisce l'adattabilità e la possibilità di riutilizzare il progetto alla luce della continua e rapida evoluzione dei dispositivi in cui può essere implementato. Si noti che anche la modalità di progettare descrivendo la rete mediante equazioni booleane si realizza nell'ambito dell'impiego di un linguaggio descrittivo. Il linguaggio descrittivo principale è il VHDL, acronimo di Vhsic (Very high speed integrated circuit) Hardware Description Language, nato agli inizi degli anni '80 nell'ambito del programma VHSIC del Dipartimento della Difesa statunitense, il cui scopo era la produzione delle nuove generazioni di circuiti integrati. Il VHDL nasce con il duplice obiettivo di avere un linguaggio con istruzioni orientate alla descrizione circuitale e di essere uno standard di interfacciamento tra vari progettisti. In un ventennio ha subito diverse fasi di 'standardizzazione', segno questo dell'effettiva efficacia del suo impiego, divenendo per esempio standard IEEE dal 1987. L'uso dei diagrammi di stato prevede, infine, la rappresentazione grafica del progetto, evidenziando gli stati stabili del sistema descritto e connettendoli mediante transizioni di cui si forniscono le condizioni di attuazione.

Simulazione funzionale

La simulazione funzionale ha lo scopo di valutare l'aderenza comportamentale del circuito logico elettronicamente descritto rispetto a quello voluto. Per far ciò si stabiliscono opportuni stimoli da fornire in ingresso al sistema e se ne valuta la risposta, senza peraltro tenere conto di aspetti implementativi quali, soprattutto, i diversi ritardi di propagazione dei segnali al suo interno. Il simulatore, valutando le risposte agli stimoli di ingresso ('vettori di test' o test bench stimuli), è in grado di effettuare una prima verifica (debugging) del progetto.

Sintesi

La fase di sintesi lega il progetto descritto alla tecnologia nella quale esso verrà implementato. Infatti in questa fase viene realizzato il file per l'implementazione del sistema nel componente scelto, tenendo conto anche del contenitore (package) del circuito integrato che viene generato. Viene inoltre generato il file di programmazione per configurare il dispositivo, oltre a un database per la simulazione di

quanto risulterà fisicamente in esso implementato. Questa fase è la più complessa di tutto l'iter di sviluppo ed è, a sua volta, strutturata in una sequenza di operazioni svolte nell'ordine in cui vengono qui di seguito descritte. La filosofia del processo è di passare dal livello 'umano' al livello 'circuitale'.

- Translation. Questo passo è ancora technology independent. La descrizione del progetto viene esaminata e ne viene verificata la correttezza e la coerenza dal punto di vista logico, della connessione dei segnali e dei pin, dei nomi usati ecc. Viene inoltre prodotta in uscita una descrizione del progetto in un unico livello gerarchico, ma sempre nel dominio logico.
- Mapping. Questo modulo, sulla base di quanto generato in uscita dal traduttore, ricava una descrizione del progetto nel dominio fisico, cioè in termini di gate. Più precisamente il progetto viene letteralmente 'mappato' in blocchi che corrispondono a quelli disponibili nel dispositivo scelto per l'implementazione. Nel fare questa operazione il modulo procede alla riduzione delle parti logiche di descrizione ridondanti, ovvero di quelle che risultano non connesse o mai usate o duplicate.
- Place & Route. Queste sono due fasi nettamente distinte tra loro. Nella fase di place i blocchi in cui il progetto è stato mappato vengono fisicamente associati ai blocchi reali del circuito integrato da programmare. A tale scopo devono essere realizzate tutte le connessioni necessarie e devono essere rispettati i vincoli di timing del progetto, vincoli intrinseci e vincoli eventualmente specificati a priori dal progettista. La fase di route connette i blocchi fisici risultanti dalla fase di place. Entrambe queste operazioni sono svolte mediante complessi algoritmi basati sulla massimizzazione di funzioni obiettivo: da un lato l'ottimizzazione delle risorse impiegate e dall'altro la minimizzazione dei ritardi nell'ottica della massimizzazione della frequenza operativa del sistema implementato. Accanto a tali obiettivi si pongono gli eventuali vincoli imposti a priori dal progettista sia in termini di tempi di ritardo sia di localizzazione di risorse nel dispositivo (tipico è il caso dei vincoli sull'associazione di segnali di I/O a precisi pin del package del dispositivo);
- Bitstream generation. Viene creato un file pronto per configurare il dispositivo a immagine del risultato della precedente fase di Place & Route.

Simulazione di Timing

La simulazione di timing è del tutto analoga alla simulazione funzionale ed è condotta sui medesimi vettori di test. La differenza risiede nel fatto che, essendo condotta a valle del processo di sintesi, la simulazione di timing è in grado di verificare la funzionalità del progetto effettivamente implementato nel dispositivo, tenendo in considerazione i ritardi sui segnali introdotti dalle connessioni e dai blocchi d'elaborazione, combinatori e sequenziali. A seguito della constatazione dell'incompleta integrità funzionale del progetto, il progettista può intervenire a questo livello o tornare alla fase precedente di sintesi, promuovendo opportune azioni atte a risolvere l'inconveniente rilevato, oppure può intervenire in prima persona nell'operazione di Place & Route. In entrambi i casi, sarà necessario successivamente procedere ad un'ulteriore simulazione di verifica come il diagramma di flusso mette in evidenza. Ovviamente una modifica comunque apportata al piazzamento e alla connessione delle risorse del dispositivo richiede una nuova generazione del file bitstream di configurazione. Alla simulazione di timing si può affiancare anche l'analisi 'statica' di timing. L'analisi statica di timing è una metodologia che verifica la consistenza di tutti i percorsi dei segnali e non solo di quelli interessati dall'applicazione degli stimoli di test. Infatti, i segnali di ingresso di test potrebbero anche non sollecitare percorsi di segnale che potrebbero essere invece critici (critical paths). E' per questo che essa può mettere in evidenza problemi non messi in luce dalla simulazione di timing.

Downloading

L'operazione di downloading consiste nel 'caricamento' nella FPGA dei dati di programmazione specifici del progetto. Tali dati sono contenuti nel file bitstream. Esistono differenti modalità di programmare gli FPGA a seconda del dispositivo e della tipologia di programmazione scelta (per esempio con il dispositivo FPGA che può essere master o slave durante il caricamento e con i dati di configurazione che possono essere inviati seriali o paralleli al dispositivo). Esiste in proposito negli FPGA un set di pin di segnali espressamente dedicati alla loro configurazione. Se l'FPGA è riprogrammabile, il file di configurazione risiede all'interno del sistema in un elemento memoria (i.e. ROM o programmabile), al quale il dispositivo può essere direttamente interfacciato, oppure nella memoria gestita da un microprocessore una cui porta di I/O può essere impiegata per l'interfacciamento con l'FPGA.

Conclusioni

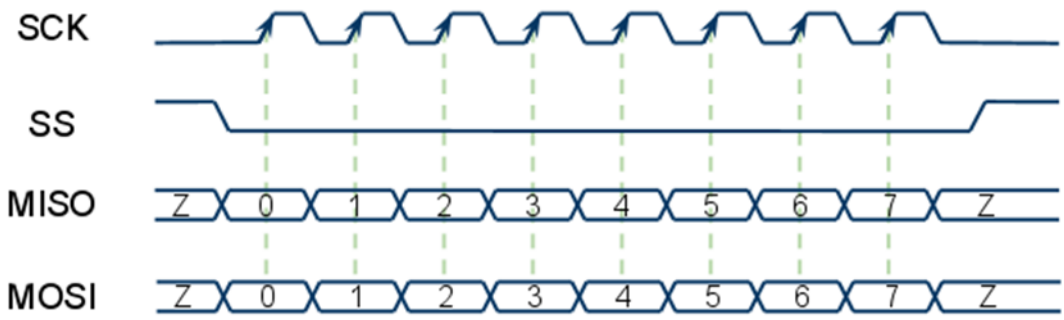
Concludendo il lavoro di tesi si è collocato nell'ambito della realizzazione di una scheda DAQ comprendendo diversi aspetti: dalla scrittura dei firmware per il processore ATMEL e per l'FPGA alla costruzione di un software per il PC capace di rendere user friendly l'interfaccia di programmazione dell'intero sistema.
Il progetto ESA è giunto alla fase di collaudo delle varie sezioni. In particolare sto partecipando quotidianamente alle fasi di test della corretta comunicazione tra ASIC, DAQ e PC.
La sorgente di segnale utilizzata in questa fase è uno strumento impulsatore, che attraverso una piccola board di switch, permette di stimolare i diversi canali selezionabili manualmente. Dalle prove effettuate si evince il perfetto funzionamento delle fasi di reset e di programmazione, mentre il processo di acquisizione necessita di ulteriori lavori di calibrazione.
Alla fine di questo lavoro seguirà l'assemblaggio completo del sistema scintillatore - matrice di SDD – ASIC – DAQ per rivelare i segnali da una sorgente di raggi gamma.
Ancora una volta in questo progetto sono emerse le potenzialità e la duttilità di un FPGA, che offre la possibilità al progettista di riconfigurare infinite volte le connessioni per raggiungere i risultati desiderati. Viste le continue evoluzioni di questo tipo di device sul mercato, e i continui progressi in termini di prestazioni e funzionalità, si è pensato di sviluppare nel futuro un sistema simile a quello descritto completamente gestito da FPGA che possa addirittura svolgere la funzione di front-end al posto di un chip dedicato (ASIC) come è avvenuto in questo progetto. Si otterrebbero ulteriori vantaggi in termini di integrazione del sistema e di costo per la realizzazione. Aspetto ancora più interessante sarebbe la possibilità di implementare il filtro shaper digitale riconfigurabile. In runtime la forma del filtro viene modificata a seconda del tipo di segnale in ingresso, questa possibilità renderebbe ancora più esteso il campo di applicazioni del sistema d'acquisizione.

Appendice A

Bus di comunicazione utilizzati

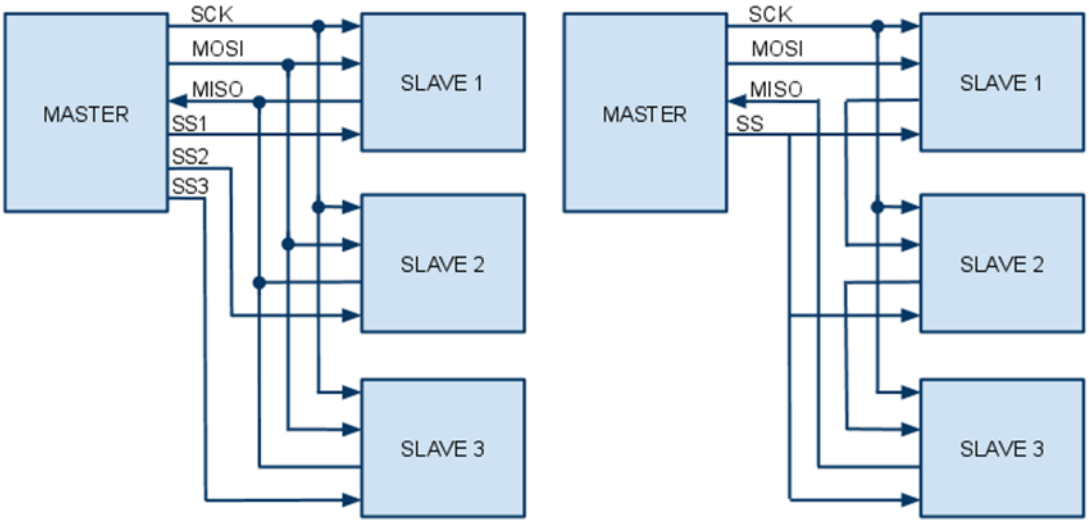
SPI (Serial Periheral Interface)

Il bus SPI è uno standard di comunicazione sincrono a quattro fili, che opera in modalità full duplex. I dispositivi comunicano in configurazione master o slave e sono ammessi slave multipli.
Una linea di Slave Select (SS) per ogni periferica, identifica il dispositivo a cui è consentito comunicare.
Una linea di Serial Clock (SCK) in uscita dal modulo master detta i sincronismi al bus.
Infine vi sono due linee dati: la Master Output, Slave Input (MOSI), utilizzata dal master per comunicare informazioni agli slave e la Master Input, Slave Output (MISO), su cui viaggiano le informazioni dagli slave al master.
Quando il bus è in stand-by il master lascia ferma la linea di clock e tiene al valore logico alto lo Slave Select. Quando deve avvenire la comunicazione, lo Slave Select del dispositivo slave prescelto viene abbassato e successivamente viene fornito il segnale di clock.
Vi sono quattro possibili modalità di funzionamento, identificate da due parametri: il CPOL e il CPHA. Se il CPOL è uguale a 0, in situazione di stand-by il clock è a valore logico basso. In questo caso, se il CPHA è uguale a 0 il dato viene letto dai due dispositivi durante il rising-edge, mentre viene imposto sulla linea durante il falling-edge. In caso contrario il dato è imposto durate il rising-edge e viene letto sul falling-edge.
In caso di CPOL uguale a 1, in situazione di riposo il clock assume valore logico alto. L'asserzione e la lettura del dato in base al CPHA è inversa rispetto al caso precedente.
In Figura 81 è mostrato, a titolo di esempio, come avviene la trasmissione nel caso in cui CPOL=0 e CPHA=0.



Appendice - Figura 1 Diagramma temporale SPI

Come si può vedere in nella figura sopra vi sono due modalità principali di interconnessione: la prima (mostrata a sinistra), molto semplice, associa uno slave select per ogni slave presente: il master comunica direttamente con la periferica prescelta. Nella seconda lo slave select è unico, la linea in uscita dal master va al primo dispositivo, quella in uscita dal dispositivo va al secondo e così via, finchè la linea in uscita dall'ultimo dispositivo torna al master. La comunicazione in questo caso è più complicata: è divisa in due fasi che si alternano ripetutamente. Durante la prima fase i moduli ricevono il dato in ingresso e comunicano in uscita il proprio dato. Nel periodo successivo si limitano a ritrasmettere il dato acquisito nella prima fase ed a ricevere, quindi, quello che era stato ricevuto dal dispositivo precedente. Il protocollo d'incapsulamento dei dati risulta quindi alquanto complesso, dovendo specificare al suo interno, quale modulo è interessato da una determinata informazione.

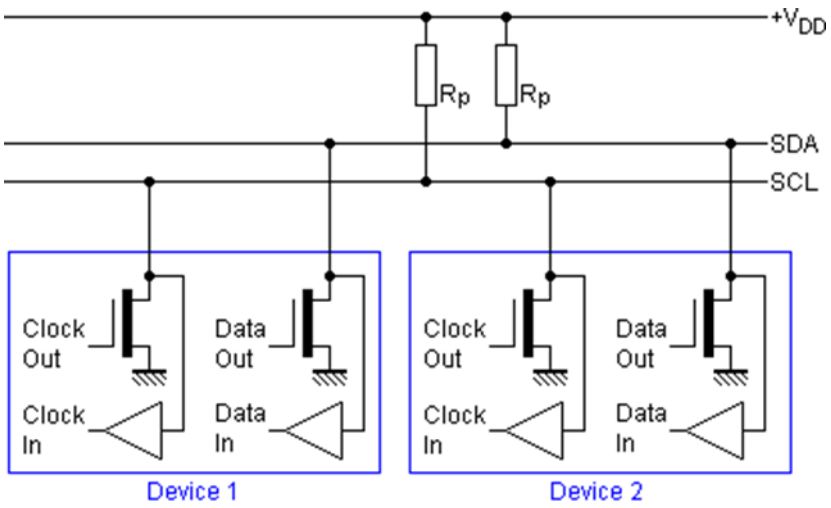


Appendice - Figura 2 Configurazione a slave indipendenti a sinistra, cooperativi a destra

I vantaggi nell'utilizzo di tale protocollo sono molteplici: innanzitutto è uno standard che ammette la trasmissione e la ricezione contemporanea, quindi con un innalzamento del throughput (rispetto al principale concorrente che può essere identificato nel protocollo I2C); inoltre, è molto versatile e consente di trasmettere parole di qualsivoglia lunghezza e contenuto; è molto semplice da implementare e ha un consumo ridotto, in quanto non necessita di complicati circuiti di ricezione; infine non è necessario fornire un indirizzo univoco a ogni periferica che viene connessa al bus.
Gli svantaggi sono invece limitati sostanzialmente all'uso di quattro linee, al posto delle 2 presenti nell'I2C e a una minore flessibilità in termini di sistema: l'aggiunta di una periferica, infatti, comporta l'aggiunta di una linea di slave select, non necessario quando ad essere utilizzato è il suo antagonista.

I2C (Inter Integrated Circuit)

L'I2C è uno standard di comunicazione seriale a due linee. Ogni componente che si connette al bus ha un indirizzo unico. Il dispositivo che incomincia la comunicazione è il master, che si occupa di gestire il segnale di clock. La struttura del protocollo permette di creare una struttura multimaster in cui, se la linea non è già utilizzata per un'altra trasmissione, ogni dispositivo può prenderne il controllo e diventare master. Generalmente, però, questo bus di comunicazione viene usato con un master solo e più slave.



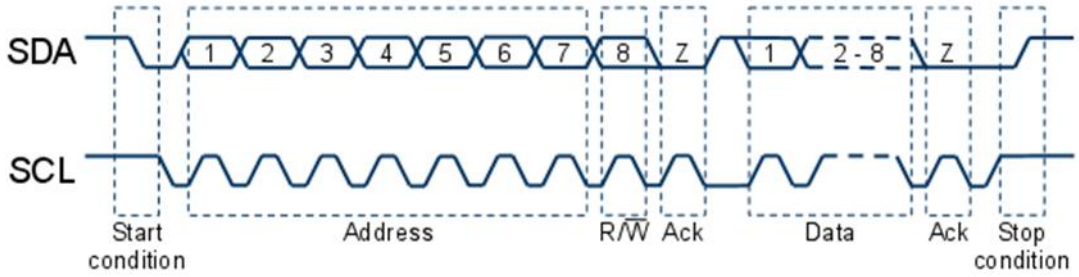
Appendice - Figura 3 Schema connessioni I2C

Lo standard definisce anche la velocità di trasmissione, che può assumere 3 valori:

- 100kbps Standard Mode
- 400kbps Fast Mode
- 3.4Mbps High Speed Mode

Le due linee, una di dato e una di clock, hanno un pull-up resistivo, quindi a riposo entrambe le linee sono alla tensione di alimentazione.
Quando il master vuole iniziare a comunicare, impone sulla linea la start condition, che consiste nell'imporre lo 0 sulla linea dati mentre la linea di clock è ancora a 1. Questa condizione, come quella di stop, è univocamente identificabile, in quanto durante la trasmissione dei dati, le transizioni avvengono sempre quando il clock è a livello logico basso.
Successivamente alla condizione di start, viene abbassata anche la linea di clock e imposto il primo bit sul canale dati. Ogni qualvolta il clock va a livello logico basso viene posto un bit sul bus dal master. Il bit viene letto dagli slave quando il clock è alto. Il primo byte, trasmesso sempre dal master, è composto da 7 bit di indirizzo e 1 bit che indica al dispositivo indirizzato se l'operazione che si vuole svolgere è di lettura (1) o di scrittura (0).
Al colpo di clock successivo la trasmissione dell'ottavo bit, il master mette la linea dati in tristate. Il dispositivo che si riconosce nell'indirizzo trasmesso, a questo punto, impone lo zero sulla linea dati. Questo bit si chiama di acknowledge e indica al master che la trasmissione è andata a buon fine. I moduli che non si riconoscono

nell'indirizzo trasmesso dal master, invece, aspettano senza fare nulla fino alla ricezione di una condizione di stop.
Al successivo colpo di clock inizia la trasmissione dei dati, o da parte del master o da parte dello slave, in base al bit di read/write. Il clock invece è sempre imposto dal master. Dopo ogni 8 bit vi è sempre un colpo di clock dedicato all'acknowledge, in modo da identificare eventuali errori di trasmissione. Il numero di byte che si possono trasmettere è totalmente arbitrario.
Quando la trasmissione deve concludersi il master impone la condizione di stop, che consiste nell'alzare il bus dati mentre il clock è alto. Lo schema in Figura 84 può aiutare visivamente a capire il funzionamento del protocollo.



Appendice - Figura 4 Diagramma temporale bus I2C

Il vantaggio principale, nell'uso di questo protocollo, sta nella struttura hardware molto semplice. Con solo due linee, si possono interfacciare fino a 128 componenti, sia master che slave. Inoltre, l'aggiunta di una periferica sulla linea non comporta nessuna modifica al resto della rete di comunicazione, il che lo rende un protocollo particolarmente flessibile, molto comodo soprattutto in fase di prototipazione.
Gli svantaggi principali, invece, sono la procedura di comunicazione abbastanza complessa e il non poter effettuare comunicazioni full duplex.

Appendice B

Modulo del kernel - Driver per l'interfacciamento con il bus EBI

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/io.h>
#include <linux/fs.h>
#include <asm/uaccess.h>
#define SUCCESS 0
#define BUF_LEN 80

#define DEVICE_NAME "testdev"
#include "testmodule.h"

/*
 * The message the device will give when asked
 */
static char Message[BUF_LEN];

/*
 * How far did the process reading the message get?
 * Useful if the message is larger than the size of the
 * buffer we get to fill in device_read.
 */
static char *Message_Ptr;

char *phy;
unsigned char *kernel_buffer;
int buffer_pos;
#define SMC_REGS_BASE 0xFFF03400

static int Device_Open = 0;
int contatore;

static int device_open(struct inode *inode, struct file *file)
{

        // Controlla se il device è già aperto
        if (Device_Open)
                return -EBUSY;

        Device_Open++;

        //Setta a zero la posizione di lettura della memoria sul FPGA
        buffer_pos = 0;
        try_module_get(THIS_MODULE);
        return SUCCESS;
}

static int device_release(struct inode *inode, struct file *file)
{
```

```
	// Chiudi il driver
	Device_Open--;

	module_put(THIS_MODULE);
	return SUCCESS;
}


// Call back invocate dal kernel quando lo user richiede un frame
static ssize_t device_read(struct file *file,	/* see include/linux/fs.h   */
			   char __user * buffer,	/* buffer to be
							 * filled with data */
			   size_t length,	/* length of the buffer     */
			   loff_t * offset)
{
	int ln_op;
	int i,j;
//	Se ho già trasmesso tutto il pacchetto ritorno 0 altrimenti continuo a leggere

	if (buffer_pos >= 16000-1)
		return 0;

	ln_op = 16000-buffer_pos;
	if (length<ln_op)
		ln_op = length;


//	Passa a livello user la memoria fisica letta dal FPGA
	copy_to_user(buffer,&kernel_buffer[buffer_pos], ln_op);

	buffer_pos += ln_op;

	return ln_op;


/*
 *Gestione delle funzioni IOCTL
 */
int device_ioctl(struct inode *inode,	/* see include/linux/fs.h */
		 struct file *file,	/* ditto */
		 unsigned int ioctl_num,	/* number and param for ioctl */
		 unsigned long ioctl_param)
{
	int i;
	int j;
	char *temp;
	char ch;

	/*
	 * Servi l'IOCTL in funzione del codice
	 */
	switch (ioctl_num) {

	// Servi la funzione get frame
	case IOCTL_GET_FRAME:
		if (phy[3] == 1)
		{
			buffer_pos=0;
			//Copia dalla memoria del FPGA nella memoria kerner
```

```
                memcpy(kernel_buffer,phy+4,16000*sizeof(unsigned char));

                return 1;
        }
        else
        {
                //Frame non disponibile
                return 0;
        }


    case IOCTL_AVAILABLE_FRAME:
        if (phy[3] == 1)
        {
                //Sono disponibili frames
                return 1;
        }
        else
        {
                //Non sono disponibile frame
                return 0;
        }

    case IOCTL_SET_MSG:
        temp = (char *)ioctl_param;
        get_user(ch, temp);
        for (i = 0; ch && i < BUF_LEN; i++, temp++)
                get_user(ch, temp);

        device_write(file, (char *)ioctl_param, i, 0);
        break;

    case IOCTL_GET_MSG:
        i = device_read(file, (char *)ioctl_param, 99, 0);
        put_user('\0', (char *)ioctl_param + i);
        break;

    case IOCTL_GET_NTH_BYTE:
        return Message[ioctl_param];
        break;
    }

    return SUCCESS;
}

/* Module Declarations */

struct file_operations Fops = {
    .read = device_read,
    .write = device_write,
    .ioctl = device_ioctl,
    .open = device_open,
    .release = device_release,      /* a.k.a. close */
};
```

```
// Punto di ingresso del modulo
int init_module()
{
        int ret_val;
        /*
         * Register the character device (atleast try)
         */
        ret_val = register_chrdev(MAJOR_NUM, DEVICE_NAME, &Fops);

        if (ret_val < 0) {
                printk(KERN_ALERT "%s failed with %d\n",
                      "Sorry, registering the character device ", ret_val);
                return ret_val;
        }

        printk(KERN_INFO "%s The major device number is %d.\n",
              "Registeration is a success", MAJOR_NUM);
        printk(KERN_INFO "If you want to talk to the device driver,\n");
        printk(KERN_INFO "you'll have to create a device file. \n");
        printk(KERN_INFO "We suggest you use:\n");
        printk(KERN_INFO   "mknod   %s   c   %d   0\n",   DEVICE_FILE_NAME,
MAJOR_NUM);
        printk(KERN_INFO "The device file name is important, because\n");
        printk(KERN_INFO "the ioctl program assumes that's the\n");
        printk(KERN_INFO "file you'll use.\n");

        printk("Mapping ESA board: ");
        phy = ioremap_nocache(0x08000000,0xFFFF);
        printk("0x%08X\n",phy);

        printk("Checking ESA board:                                    ");
        if ((phy[0] != 0x55) && (phy[1] != 0xAB))
        {
                printk("[FAILED]\n");
                return -1;
        }
        printk("[  OK  ]\n");

        printk("Allocating kernel frame buffer....                ");

        kernel_buffer = kmalloc(16000*sizeof(unsigned char),GFP_KERNEL);

        if (kernel_buffer == NULL)
        {
                printk("[FAILED]\n");
                return -1;
        }
        else
        {
                printk("[  OK  ]\n");
        }
contatore = 0;
        return 0;

}

/*
 * Cleanup – funzione per la de-registrazione del modulo
```

```
 */
void cleanup_module()
{
        int ret;
        unregister_chrdev(MAJOR_NUM, DEVICE_NAME);

}
```

Command server - Interfaccia per il controllo del register file nel dispositivo FPGA

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>

#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>

#include <linux/types.h>
#include </home/andrea/linux-2.6.25.10.atmel.2/include/linux/spi/spidev.h>

#include <time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include <signal.h>
#include <unistd.h>             /* exit */


int sockfdSM;           //SOCKET PORTA STREAM


void dostuff(int);
char *bufferout;


static int verbose;

int kbhit(void)
{
        struct timeval tv;
        fd_set read_fd;

        tv.tv_sec=0;
        tv.tv_usec=0;
        FD_ZERO(&read_fd);
        FD_SET(0,&read_fd);

        if(select(1, &read_fd, NULL, NULL, &tv) == -1)
```

```
                return 0;

        if(FD_ISSET(0,&read_fd))
                return 1;

        return 0;
}


static void pabort(const char *s)
{
        perror(s);
        abort();
}


// Leggi i dati relative al controller SPI

static void dumpstat(const char *name, int fd)
{
        __u8    mode, lsb, bits;
        __u32   speed;

        if (ioctl(fd, SPI_IOC_RD_MODE, &mode) < 0) {
                perror("SPI rd_mode");
                return;
        }
        if (ioctl(fd, SPI_IOC_RD_LSB_FIRST, &lsb) < 0) {
                perror("SPI rd_lsb_fist");
                return;
        }
        if (ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits) < 0) {
                perror("SPI bits_per_word");
                return;
        }
        if (ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed) < 0) {
                perror("SPI max_speed_hz");
                return;
        }

        printf("%s: spi mode %d, %d bits %sper word, %d Hz max\n",
                name, mode, bits, lsb ? "(lsb first) " : "", speed);
}



//Esegui operazioni di comunicazione PC - SPI
void dostuff (int sock)
{
        int n=0;
        int j;
        int cycles=0;
        int bytecount =0;
        int TXbytecount =0;
        int frame_pixels[100];
        int packet_counter=0;
        unsigned char buffer[256];
        int number_of_frame;
        int esa_desc;
```

```
int address, data;
int ret;
static long int speed = 16000000;
struct spi_ioc_transfer         xfer[2];
unsigned char           buf1[64],buf2[64], *bp;
int                     status;
int                     i;
unsigned int   ADDRESS,DATA;
int             fd;
unsigned int *sample;

fd = open("/dev/spidev1.0", O_RDWR);
if (fd < 0) {
        perror("open");
        return 1;
}

/*
 * Imposta i parametric come la velocità di comunicazione
 */
ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
if (ret == -1)
        pabort("can't set max speed hz");

ret = ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed);
if (ret == -1)
        pabort("can't get max speed hz");

dumpstat("/dev/spidev1.0", fd);


printf("Listening for commands...\n");

  n= 1;
  while(n>0)
  {
        bzero(buffer,0xFF);
        bytecount =0;
        TXbytecount = 0;

        while((n>0) && (buffer[0]!='S'))
                n = recv(sock,buffer,1,MSG_WAITALL);

        n = recv(sock,buffer,7,MSG_WAITALL);


        memset(xfer, 0, sizeof xfer);
        memset(buf2, 0, sizeof buf2);
        memset(buf1, 0, sizeof buf1);

        // Prepara il pacchetto SPI (16 bit indirizzo e 32 bit dati)

        buf1[0] = 0x00;
        buf1[1] = 0x89;
        buf1[3] = buffer[1];//(ADDRESS >> 8);
        buf1[2] = buffer[2];//(ADDRESS & 0xFF);
        buf1[7] = buffer[3];//(DATA >> 24);
        buf1[6] = buffer[4];//(DATA >> 16) & 0xFF;
        buf1[5] = buffer[5];//(DATA >> 8) & 0xFF;
```

```
                buf1[4] = buffer[6];//(DATA & 0xFF);

                xfer[0].tx_buf = (__u64) buf1;
                xfer[0].len = 8;
                xfer[0].delay_usecs = 0,

                xfer[1].rx_buf = (__u64) buf2;
                xfer[1].len = 0;
                xfer[1].delay_usecs = 0,

                // invia il pacchetto
                status = ioctl(fd, SPI_IOC_MESSAGE(2), xfer);
                if (status < 0) {
                        perror("SPI_IOC_MESSAGE");
                }

                printf("SPI Writen: %X %X %X %X %X %X\n",buf1[2],buf1[3],buf1[4],buf1[5],buf1[6],buf1[7]);


          }


        printf("Client disconnesso\n");
        close(fd);

}


// Creare il sever ethernet

int create_server()
{
    int newsockfd, portno, clilen, pid;
    char buffer[256];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    sockfdSM = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfdSM < 0)
      error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = 101;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfdSM, (struct sockaddr *) &serv_addr,
          sizeof(serv_addr)) < 0)
          error("ERROR on binding");
    printf("Waiting connection\n");
    listen(sockfdSM,5);
    clilen = sizeof(cli_addr);

    while (1) {
       newsockfd = accept(sockfdSM,
            (struct sockaddr *) &cli_addr, &clilen);
       if (newsockfd < 0)
          error("ERROR on accept");
```

```
        pid = fork();
        if (pid < 0)
            error("ERROR on fork");
        if (pid == 0)  {
            close(sockfdSM);
            dostuff(newsockfd);
            exit(0);
        }
        else close(newsockfd);

    } /* end of while */
    return 0;
    return 0;
}



int main(int argc, char **argv)
{

        int pid;

        printf("Starting SPI Server Application...\n");
        pid = fork();
      if (pid < 0)
            error("ERROR on server fork");
        if (pid == 0)  {
                    create_server();
        }
        else {

                    waitpid(pid,NULL,0);
        }

        return 0;
}
```

Bibliografia

- Abba A. (2010) Methodology and Advanced Electronic Architectures for High Performance Digital Processing. Tema maggiore di dottorato.

- Howard W. Johnson, Martin Graham (1993). High-speed Digital Design.

- Peter J. Ashenden (1990). The VHDL cookbook.

- Morgan Kaufmann Publishers (1990). The Designer's Guide to VHDL.

- C.E.Spurgeon. (2000). Ethernet: The definitive guide.

- Xilinx. (2010). Spartan 3A Datasheet. Tratto da Sito Web Xilinx: http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf

- Philips Semiconductors. (2000). The I2C-BUS Specification. Tratto da Sito Web NXP: http://www.nxp.com/acrobat_download2/literature/9398/39340011.pdf

- Zappa F. (2010). Sistemi Elettronici. Milano: Casa Editrice Esculapio.