POLITECNICO DI MILANO

Facoltà di Ingegneria dell'Informazione

Corso di Laurea Magistrale in
Ingegneria Elettronica

Sviluppo di un emulatore multicanale di gamma camera

Relatore: Prof. Angelo GERACI
Correlatore: Ing. Andrea ABBA

Tesi di Laurea Magistrale di:
Prospero LOMBARDI
Matr. 783939

Anno Accademico 2012 - 2013

.

Alla mia famiglia ed in particolar modo alla mia fidanzata,
il loro aiuto e sostegno sono stati fondamentali.

Indice

Elenco delle figure

.

Sommario

L'imaging diagnostico riveste, al giorno d'oggi, un ruolo centrale nella tecnica medica. Diverse sono le indagini che possono essere condotte ed ognuna di queste ha vantaggi e svantaggi dal punto di vista dell'incolumità del paziente e della qualità dell'informazione che è in grado di fornire. Le tecniche che permettono di investigare l'evoluzione metabolica cellulare, ad esempio per lo studio dello sviluppo tumorale, consistono nell'impiego di radiazione gamma localizzata nella zona sotto osservazione e di una sofisticata elettronica di rivelazione. Per avere immagini ad alta qualità è di fondamentale importanza avere sistemi di rivelazione sempre più innovativi e performanti che supportino i medici nel loro lavoro.

L'obiettivo del progetto è quello di realizzare un dispositivo che permetta il testing ed il debugging di macchine di imaging nucleare. In particolare, attraverso l'emulatore multicanale, è possibile andare a verificare il corretto funzionamento dell'elettronica di ricostruzione dell'immagine, interfacciata con l'elettronica di rivelazione della radiazione, senza ricorrere alle pericolose sorgenti radioattive.

Questo è reso possibile attraverso una sofisticata tecnica di emulazione che permette di ricostruire i parametri fondamentali che accompagnano la radiazione, quali ad esempio le statistiche energetica e temporale di emissione.

L'emulatore multicanale si propone di sostituirsi sia alla fase di generazione, all'interno dei tessuti del paziente, che a quella di rilevazione vera e propria, usualmente attuata dalla *gamma camera,* al fine di produrre dei segnali elettrici confondibili con quelli di un esperimento reale.

Ad oggi è già possibile emulare singole sorgenti radioattive per cui il con-

tributo che questo progetto apporta nell'ambito dell'emulazione della radiazione è la possibilità di produrre un emulazione di tipo matriciale; l'emulatore viene programmato con un'immagine di partenza arbitraria e da questa produce i segnali corretti che ne permettono la ricostruzione successiva.

Oltre a descrivere dettagliatamente gli algoritmi di emulazione si presenterà la struttura hardware realizzata che ne permette l'esecuzione ottimale.

Abstract

Nowadays diagnostic imaging is widely used into medical practice. There are different ways in order to make good imaging tests, but, of course, each one offers advantages and disadvantages related to patient's health. Gamma rays, produced by nuclear decay induced into human tissutes, are used to investigate cellular metabolic activity, i.e if there is high metabolic activity the area under investigation is probably a cancer. This is performed through sofisticate detection machineries that are requested to be as efficient and noiseless as possible to give detailed images supporting medic diagnosis.

Task of this project is to realize an electronic equipement able to operate as a testing and debugging device for nuclear imaging detection systems. Through the multichannel emulator we can see if the reconstruction electronic connected to the gamma detector is really working properly, without using dangerous radioactive real sources.

This is possible thanks to an advanced emulating algorithm which is able to simulate the most relevant radiation parameters such as energetic and temporal emission statistics.

Gamma generation, within patient tissutes, and their revelation, tipically realized with a *gamma camera*, are totally replaced with the multichannel emulator which grant aliasable electric signals with respect to real experiment ones.

Single radioactive sources have been properly emulated until now so this project, speaking of radiation emulation, offers the new, never sperimented, matrix emulation; the emulator is programmed with an arbitrary start image and produces electric signals necessary for the successive reconstruction phase.

Here we will present either the computational algorithms used and the hardware structure built in such a way that it's able to exploit the optimal execution of the algorithms.

Capitolo 1

Introduzione all'imaging medico

Attualmente le molteplici tecniche di imaging utilizzate in medicina consentono di analizzare l'interno del corpo umano con precisione almeno millimetrica. La tecnica radiografica è stata una delle prime metodologie impiegate a tale scopo ed ha permesso lo studio di quasi tutti gli organi del corpo umano con o senza l'ausilio di mezzi di contrasto ovvero sostanze in grado di modificare il modo in cui una particolare regione del corpo del paziente sotto esame appare in un'immagine medica. E' noto che l'impiego di radiazione X può comportare la rottura dei legami molecolari sui tessuti colpiti in ragione dell'elevata energia che la caratterizza. Questo porta alla formazione di ioni e radicali liberi che interagiscono con altri atomi provocando danni, a volte irreversibili, che conducono alla morte cellulare. Ciò ha chiaramente impatto negativo sulle funzionalità dell'organo cui la cellula appartiene e, quindi, anche sull'intero individuo. Nonostante le controindicazioni, tale tecnica radiografica ancora oggi risulta ampiamente diffusa e sembra rimarrà tale almeno nel breve termine in quanto è ormai consolidata e può essere fruita ad un costo relativamente basso rispetto ad altre tecniche più sofisticate. Esempi di comuni esami che sfruttano la radiazione X sono la mammografia, la radiografia dello scheletro e l'angiografia, ovvero la tecnica che permette un'osservazione selettiva dei vasi sanguigni. Negli ultimi anni innovazioni tecnologiche hanno permesso l'impiego di strumenti meno invasivi per l'analisi dell'interno del corpo umano. In contrapposizione alle immagini bidimensionali offerte da una tipica lastra a raggi X, dove viene mostrato in un singolo

piano tutto lo spessore dell'oggetto in esame, oggi è possibile godere di una rappresentazione a "fette", o meglio tomografica, della zona d'interesse. La tomografia computerizzata (CT) può essere utilizzata in qualsiasi distretto corporeo ed è generalmente capace di visualizzare con sufficiente completezza lo scheletro e le articolazioni senza necessità di far ricorso al mezzo di contrasto.

1.1 Immagini CT

Le immagini CT sono rappresentative della distribuzione del coefficiente di attenuazione dei raggi X che attraversano un campione biologico. In esse sono osservabili dettagli anatomici non apprezzabili con altre tecniche. In forza di tale proprietà la CT ha raggiunto rapidamente una vasta diffusione nella pratica clinica.

Un sistema CT è costituito dai tubi radiogeni, dal tomografo propriamente detto (scanner), da un sistema computerizzato di calcolo, dotato di processori ausiliari dedicati per ridurre (fino a meno di 1 secondo per immagine) i tempi di calcolo richiesti dagli algoritmi ricostruttivi, e da una stazione di consultazione, dotata di display e di dispositivi interattivi che consentono la manipolazione delle immagini. La stazione di consultazione deve consentire all'utilizzatore di estrarre l'informazione disponibile nelle varie procedure diagnostiche. Genericamente, il sistema tubo radiogeno - tomografo ruota solidalmente effettuando scansioni successive durante la rotazione. Queste scansioni generano dati che vengono raccolti in matrici e rielaborati al fine di ottenere un'immagine interpretabile.

Un tessuto attraversato da radiazione X esibisce un coefficiente di attenuazione ρ con la seguente caratteristica; maggiore è la sua densità maggiore sarà l'attenuazione subita dai raggi nell'attraversarlo, e, quindi, minore sarà la quantità di essi che raggiungerà lo scanner. La tonalità dell'immagine ottenuta, in termini di livello nella scala di grigi, sarà indice di questa attenuazione.

Nell'esempio di figura 1.1 si analizza un disco di densità uniforme all'interno del quale ne vengono inseriti uno di densità inferiore e uno di densità

maggiore rispetto al primo. I profili di attenuazione lungo gli assi orizzontale e verticale sono mostrati accanto ad esso. Si notano dei "picchi" in corrispondenza del disco di elevata densità e delle "valli" nel caso opposto. E' importante osservare che il profilo della proiezione è diverso per i due assi per via del differente punto di osservazione. Se si osservano le proiezioni lungo tutte le direzioni e non soltanto le due mostrate in questo esempio è possibile ottenere abbastanza contenuto informativo per riuscire a ricostruire l'immagine 2D dell'oggetto.

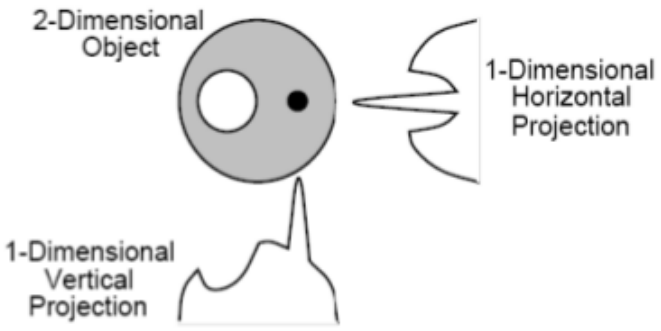

Figura 1.1: *Profili di attenuazione.*

I tomografi computerizzati sono probabilmente le apparecchiature digitali più utilizzate nella diagnostica medica per analisi a costi relativamente ridotti e, per questo, quasi tutti gli ospedali ne sono dotati. Il risultato di un esame CT è una serie di matrici (*slices*), in media 30-40, allineate perpendicolarmente all'asse definito dalla spina dorsale del paziente. Ogni *slice* rappresenta una fetta del corpo del paziente di un determinato spessore (1-10mm). Per la maggior parte dei tomografi la risoluzione ottenibile per ciascuna matrice è di 256 o 512 pixel. Ciascun pixel rappresenta le caratteristiche di assorbimento dei raggi X da parte di un piccolo volume del corpo umano individuato dai limiti fisici del pixel stesso. Il numero di *slices* varia in funzione della distanza tra le sezioni e dell'estensione dell'organo da esaminare. I moderni tomografi riescono ad acquisire una *slice* in circa un secondo e quindi l'intero esame richiede un tempo che va dai 5 ai 15 minuti; la dose di radiazioni da CT è paragonabile con quella di una serie di tradizionali lastre a raggi X.

Si tenga presente che per aumentare la risoluzione dell'immagine, il paziente viene sottoposto a dosi crescenti di radiazione e la durata stessa dell'esame incrementa. Questo porta a rischi per il paziente in termini di dose assorbita. In più la qualità dell'immagine può degradare, in quanto, per tempi di esposizione lunghi, il paziente si può muovere; un fattore di rumore può essere il respiro o il battito cardiaco.

La strumentazione di misura, quella che viene definita normalmente scanner, gioca un ruolo essenziale nell'imaging ricostruttivo. L'evoluzione più appariscente nella strumentazione CT è quella relativa alla geometria di scansione finalizzata all'aumento della velocità di acquisizione. Tutte le geometrie di scansione adottate nelle varie generazioni CT comprendono una sorgente di radiazione costituita da un tubo a raggi X e un sistema di rivelazione basato su sensori di vario tipo: sia sorgente che rivelatori effettuano movimenti rispetto all'oggetto per generare un insieme di proiezioni della sezione in esame.

Si distinguono quattro generazioni di scanner:

- **Prima generazione:** Il sistema è costituito dalla sorgente e da un singolo rivelatore. Il fascio di radiazione è collimato a pennello. L'acquisizione di una proiezione viene effettuata traslando tubo e rivelatore sull'asse trasversale mentre, per cambiare l'angolo di proiezione, è necessaria una loro rotazione. Tale circostanza implica un lungo tempo di acquisizione.

- **Seconda generazione:** Sono presenti una decina di rivelatori allineati lungo una retta. La radiazione viene collimata a pennello per ogni sensore. Il funzionamento è analogo a quello della prima generazione ma il tempo di scansione risulta dieci volte minore.

- **Terza generazione:** Viene introdotto un array di un migliaio di rivelatori disposti ad arco e capaci di ruotare solidalmente con la sorgente intorno all'oggetto.

- **Quarta generazione:** Si ha un unico anello fisso di rivelatori. L'unico elemento in rotazione è il tubo radiogeno. Quest'ultima categoria di scanner offre le prestazioni migliori per tempo di acquisizione.

1.2 Imaging radioisotopico

La medicina nucleare si serve della radioattività per scopi diagnostici o terapeutici, utilizzando radiofarmaci per evidenziare zone ad alta o bassa attività metabolica. Il radiofarmaco può essere introdotto per iniezione, inalazione o ingestione. Anche se si serve di radiazioni ionizzanti come la tecnica radiografica, la medicina nucleare presenta differenze sostanziali rispetto alla radiologia.

Mentre nelle tecniche a raggi X si conosce la localizzazione sia della sorgente che del sensore, nelle tecniche ad emissione di radiazione è conosciuta solo la posizione del sensore: serve, quindi, un dispositivo detto collimatore per ottenere un'immagine della distribuzione spazio-temporale della radioattività nel paziente. Un'altra differenza importante rispetto alla tecnica radiografica è che mentre l'irraggiamento X, concentrato sulle strutture di interesse, ha la durata determinata dalle modalità dell'esame, l'irraggiamento dei composti radioattivi che si diffondono nel corpo del paziente, ha una durata dipendente dalla loro *emivita* e dal loro tempo di fuoriuscita.

In generale le misure di raggi X sono di tipo integrale, effettuate su flussi fotonici relativamente intensi; al contrario le misure in medicina nucleare sono a conteggio diretto, effettuate su flussi fotonici sufficientemente bassi da consentire la registrazione dei singoli fotoni che arrivano al sensore. Di conseguenza le immagini di raggi X sono a statistica più alta e più adatte a visualizzare la forma e i piccoli dettagli di una struttura.

Esistono due tipologie diagnostiche in grado di effettuare rivelazione di radiazione gamma: la scintigrafia planare (SPECT), e la tomografia a emissione di positroni (PET) che forniscono una immagine bidimensionale della distribuzione di radioattività nel soggetto.

1.3 SPECT

Quasi ogni sistema esistente di imaging medico nucleare, è basato sull'architettura della gamma camera ideata nel 1958 da Hal O. Anger. Il suo progetto originale è ancora ampiamente utilizzato.

In una Anger camera convenzionale, la rivelazione di fotoni gamma e la derivazione della loro posizione di generazione, sono basate su una conversione di energia a due stadi, preceduta dalla selezione dei fotoni che viaggiano lungo una ben definita direzione rispetto al piano di ingresso del sistema di rivelazione grazie ad un *collimatore*.

Anche nella versione classica di SPECT (*Single Photon Emission Computed Tomography*) viene utilizzata una gamma camera per la rivelazione dei fotoni gamma. In particolare si ricorre ad una struttura ad hoc coinvolgente un insieme di semplici gamma camera, tipicamente tre, in grado di compiere una rotazione di 360 gradi intorno al punto che si vuole osservare. La figura 1.2 illustra quanto segue.

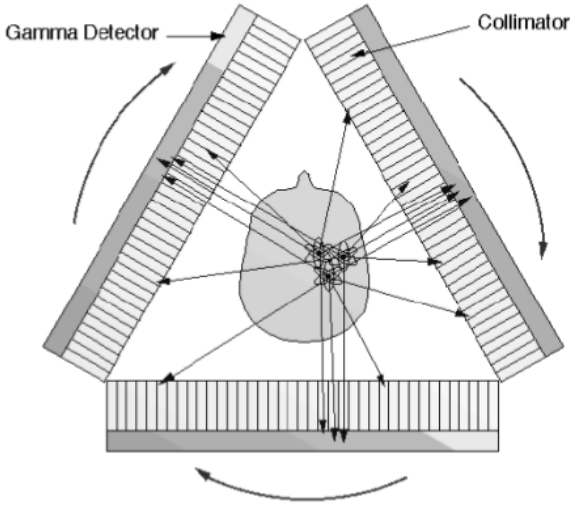

Figura 1.2: *Applicazione della gamma camera nella SPECT.*

Le gamma camera, accoppiate opportunamente con dei collimatori, effettuano un *campionamento angolare* della distribuzione tridimensionale del radiofarmaco. Poiché una gamma camera è in grado di acquisire in un solo

colpo un'immagine si ha che ogni riga di questa si riferisce ad una slice tomografica ed il numero di righe indica il numero di slice che è possibile acquisire in un esame. In ciascuna proiezione le diverse righe della matrice rappresentano, dunque, profili di radioattività corrispondenti alle diverse sezioni tomografiche. Ogni punto del profilo può essere pensato come l'integrale di linea (se trascuriamo l'attenuazione introdotta dal corpo del paziente) della radioattività osservata lungo la traiettoria perpendicolare al piano della gamma camera e passante per il punto stesso. Al termine della rotazione pertanto si sono ottenuti i dati di campionamento angolare e lineare che consentono la ricostruzione di diverse sezioni. Il numero di posizioni angolari e di colonne della matrice di acquisizione determinano, insieme al collimatore, la risoluzione spaziale di ciascuna fetta ricostruita.

Un tipico studio SPECT prevede, ad esempio, l'acquisizione di 64 proiezioni (5.6° di campionamento angolare) in matrici 128x128. L'evoluzione della tecnica SPECT ha portato a sistemi dotati di più *teste* di rivelazione al fine di aumentare l'efficienza di raccolta dei gamma emessi, consentendo la riduzione del tempo di acquisizione.

E' importante sottolineare che a differenza delle tecniche tomografiche a raggi X la SPECT, come è tipico della medicina nucleare, è capace di visualizzare la funzionalità di un organo piuttosto che la sua forma come ad esempio la funzione renale, la ventilazione polmonare e la visualizzazione di masse tumorali. Esistono punti caldi, ovvero ad alta attività metabolica, come un tumore, oppure punti freddi, zone patologiche.

1.4 Gamma camera

Nel progetto originale di H.O. Anger, solo i fotoni gamma aventi direzione ortogonale al piano di ingresso del sistema, vengono rilevati. Nel primo livello di conversione, l'energia di ogni fotone gamma, filtrato selettivamente dal collimatore, viene utilizzata per generare alcune migliaia di fotoni ottici per mezzo dell'utilizzo di un cristallo scintillatore in ioduro di sodio (NaI). Nel secondo livello di conversione energetica , un array di tubi fotomoltiplicatori (PMT) converte l'energia dei fotoni ottici in segnali elettrici. Le ampiezze di

picco dei segnali elettrici provenienti dai PMT per ogni evento di generazione ottica, vengono misurate da un'elettronica dedicata. La distribuzione delle ampiezze di picco è ,dunque, usata per ricostruire le coordinate x, y di interazione sul piano definito dal cristallo scintillatore. La somma delle ampiezze di picco, d'altro canto, può essere utilizzata per ricostruire l'energia rilasciata nel punto di interazione. Il complesso degli eventi di interazione costituisce un'immagine che tipicamente viene visualizzata su un display. Ad ogni pixel dell'immagine è associata una particolare coordinata (x_1, y_1); L' intensità di colore rappresenta il numero di eventi ricostruiti in quella particolare posizione.

L'architettura proposta da Anger è rimasta invariata per anni e lo ioduro di sodio che compone il cristallo scintillatore così come i PMT sono ancora ampiamente diffusi nell'ambito dell'imaging nucleare nonché nei contesti di diagnostica medica. Un buon numero di varianti sono state adottate dal primo prototipo di γ camera, comunque nel seguito si analizzeranno le componenti dello schema classico con qualche variante per ciò che concerne i detector.

1.4.1 Parametri fondamentali della gamma camera

Le specifiche rilevanti di un sistema di imaging finalizzato per scopi di medicina nucleare o in ambito di ricerca sono: Risoluzione spaziale intrinseca, Risoluzione spaziale del sistema, linearità spaziale del sistema, risoluzione energetica intrinseca, risoluzione energetica del sistema, efficienza geometrica intrinseca e sensitività del sistema, campo di osservazione (*Field Of View FOV*) , uniformità di campo, capacità di conteggio eventi.

Risoluzione spaziale. E' la distanza risolvibile tra due punti sorgenti di radiazione. La risoluzione spaziale intrinseca viene definita per un sistema di imaging senza collimatore. Dunque i parametri che caratterizzano questa figura di merito sono il tipo di materiale di cui è costituito il cristallo scintillatore ed il suo spessore, il numero e la grandezza dei fotodetector, la loro efficienza di collezione e le caratteristiche dell'accoppiamento tra cristallo e fotodetector. La

risoluzione spaziale del sistema caratterizza l'apparato di imaging con il collimatore ed è determinata combinando la risoluzione intrinseca con la risoluzione del collimatore che dipende dalle sue caratteristiche costruttive e dalla distanza di osservazione. La risoluzione spaziale è di notevole importanza nella strumentazione nucleare poiché fornisce un limite inferiore nel poter distinguere ad esempio piccole masse tumorali vicine.

Risoluzione energetica. Questa grandezza caratterizza l'abilità del sistema di identificare accuratamente gli eventi di fotopicco e, quindi, di distinguere i fotoni gamma primari ovvero quelli che non hanno subito fenomeni di scattering da quelli secondari. La risoluzione energetica intrinseca R_{ei} è la somma di varie componenti derivanti dalla statistica caratterizzante il processo fisico di rivelazione R_{Estat} , dalle proprietà dello scintillatore R_{Esci} e dalle caratteristiche e del fotodetector e dell'elettronica di elaborazione R_{Eelect}: $R_{Ei} = \sqrt{R_{Estat}^2 + R_{Esci}^2 + R_{Eelect}^2}$. R_{Ei} varia in funzione dell'energia del raggio gamma e viene tipicamente misurata con la *larghezza a metà altezza* FWHM della curva gaussiana rappresentante lo spettro di distribuzione di energia dei raggi. La risoluzione energetica del sistema impone un vincolo sulla capacità di ottenere immagini accurate rappresentanti la distribuzione di radiofarmaco localizzato in profondità all'interno del corpo del paziente. E' importante osservare, infatti, che una porzione significativa dei raggi gamma, generati in distretti profondi, lungo il percorso verso la gamma camera, subisce interazioni con i tessuti. La maggior parte di queste sono di tipo anelastico (scattering Compton) e fanno perdere l'informazione relativa al punto di generazione del raggio alterandone la traiettoria. I fotoni gamma scatterati la cui energia è diversa da quelli primari rappresentano solo un fondo di rumore nell'immagine finale che può essere eliminato se il sistema di rilevazione è in grado di selezionare solo fotoni contenuti in una stretta finestra energetica.

Sensitività. Essa definisce l'efficienza del sistema di imaging nel rivelare la radiazione. Per una data quantità di radioattività, un sistema con elevata sensitività (S) fornirà più informazione nell'immagine rispetto ad uno con minore sensitività. Questo parametro viene calcolato come il rapporto tra il tasso di eventi rivelati e la dose di radiofarmaco somministrata al paziente. S dipende dal materiale di cui è composto il detector, il suo spessore e la sua finestra energetica di accettazione. Un modo equivalente per esprimere la sensitività del sistema di imaging è attraverso la sua efficienza di rivelazione. Ciò è di scarso interesse nell'ambito biomedico in quanto tiene conto della sola quantità di fotoni gamma emessi dal corpo che naturalmente sono inferiori rispetto a quelli prodotti dal radiofarmaco somministrato nel paziente.

Campo di osservazione. Esso rappresenta la regione che può essere scannerizzata dal sistema. L'uniformità di campo, invece, rappresenta la capacità del sistema, quando esposto a radiazione nella suo campo di osservazione, di produrre un'immagine uniforme.

Capacità di conteggio eventi. Dopo la rivelazione di un fotone gamma la camera non è in grado di rivelare adeguatamente un nuovo evento per un certo intervallo di tempo denominato tempo morto. Questo tempo insieme a quello necessario perché lo scintillatore esaurisca la sua luminescenza impongono il tasso massimo di eventi che il sistema può analizzare.

1.4.2 Il collimatore

Al fine di ricostruire la distribuzione di radiofarmaco all'interno dell'organo sotto esame è fondamentale che vi sia una correlazione tra la posizione di emissione del fotone gamma e le coordinate di interazione nel detector. Perché ciò sia possibile è necessario introdurre un collimatore che effettui una selezione "meccanica" dei fotoni basata sulle loro traiettorie di propagazione verso il detector. Senza un collimatore i raggi provenienti da tutte le direzioni verrebbero rivelati generando di fatto un'immagine sfuocata e dunque

inutilizzabile. Per produrre un'immagine nitida rappresentante la reale distribuzione di radiofarmaco è necessario che vengano presi in esame soltanto i raggi gamma propaganti lungo una ben definita direzione (ad esempio normale con la superficie della gamma camera). Sfortunatamente tale vincolo non può essere pienamente soddisfatto nei sistemi reali vuoi per la dimensione finita del collimatore vuoi per l'inevitabile presenza di scattering Compton all'interno dei tessuti.

Si consideri un collimatore parallelo come mostrato in figura 1.3 dove tutti

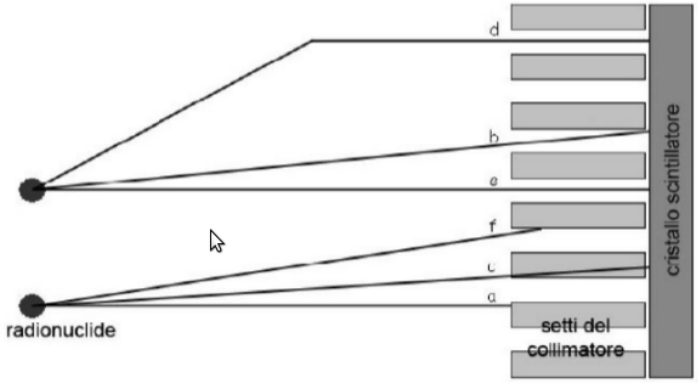

Figura 1.3: *Collimatore parallelo.*

i fori abbiano la stessa apertura ed idealmente soltanto i fotoni perpendicolari al detector possano passare inalterati. Con tale setup possono verificarsi diverse situazioni:

- I fotoni posseggono la giusta direzione ma vengono assorbiti dal setto a causa del suo spessore finito
- I fotoni non hanno la giusta direzione ma passano comunque per via della larghezza finita dei fori
- I fotoni hanno la giusta direzione ma provengono da interazioni nel materiale
- I fotoni non vengono completamente assorbiti dai setti

Tutti questi fenomeni generano errori e sfocatura nell'immagine ricostruita.

I Collimatori vengono realizzati con materiali ad alto numero atomico Z. Infatti, la probabilità che un fotone gamma venga assorbito da un certo materiale dipende da una grandezza nota come *lunghezza di assorbimento* L_{ass} che dipende dalla natura del materiale stesso; in particolare decresce in funzione del numero atomico. Dalla legge esponenziale $T = e^{-\frac{x}{L_{ass}}}$, che indica la percentuale di radiazione gamma in grado di oltrepassare uno strato di spessore x di un certo materiale, si evince che conviene utilizzare materiali con piccola lunghezza di assorbimento per far in modo che la radiazione con un angolo errato venga tempestivamente assorbita dal collimatore. In questo modo è possibile realizzare setti piccoli e fitti per ridurre alcune delle problematiche sopra elencate. I processi di costruzione impongono dei limiti nella scelta dei materiali in termini di malleabilità e rigidità. Tra i materiali di impiego comune ricordiamo il piombo e il tungsteno con numero atomico rispettivamente 82 e 74.

1.4.3 Cristallo scintillatore

L'uso di un cristallo scintillatore è necessario per convertire la radiazione gamma in radiazione nello spettro del visibile. Quasi tutte le modalità di imaging medico che richiedono la rivelazione di fotoni ad elevata energia (raggi x e γ) usano scintillatori. Le caratteristiche di un buono scintillatore sono : elevata densità e numero atomico, rigidità meccanica, robustezza alla radiazione, basso costo,lunghezza d'onda di emissione appropriata,elevata luminescenza di uscita, buona proporzionalità e breve tempo di decadimento.

Le caratteristiche di uno scintillatore possono essere raggruppate in caratteristiche fisiche o luminescenti. Tra le prime rientrano l'efficienza di rivelazione, la stabilità chimica e meccanica, forma fisica ed igroscopicità. Tra le caratteristiche luminescenti invece rientrano la lunghezza d'onda di emissione, la resa ottica, la risoluzione energetica, proporzionalità.

L'efficienza di rivelazione, una delle caratteristiche più importanti, dipende dalla capacità dello scintillatore di interagire con il fotone incidente. Alle energie dei raggi gamma i meccanismi di interazione dominanti sono l'assorbimento fotoelettrico e lo scattering Compton; risulta che l'efficienza è

maggiore per densità e numero atomico elevati. Sfortunatamente i materiali che hanno buone efficienze presentano instabilità chimica e scarsa robustezza meccanica che sono requisiti necessari nella maggior parte delle applicazioni, in particolar modo negli ambienti clinici per ragioni di sicurezza.

La lunghezza d'onda d'emissione viene specificata sul picco dello spettro di emissione della radiazione e deve essere opportunamente selezionata in accordo con la lunghezza d'onda di picco del rivelatore accoppiato per massimizzare la sensitività dello scanner. La resa ottica è la luce in uscita, espressa in numero di fotoni, per 1MeV di radiazione ionizzante assorbita. Questa caratteristica contribuisce a determinare sia la risoluzione spaziale che la risoluzione in energia del rivelatore. Il tempo durante il quale il bagliore prodotto dal raggio gamma incidente persiste nel cristallo determina un vincolo sul tempo necessario allo scintillatore per essere sensibile alla radiazione ionizzante; questa proprietà costituisce il fattore dominante per il conteggio di eventi dello scanner. La presenza di componenti ritardate nella risposta dello scintillatore dovrebbe essere evitata in quasi tutte le applicazioni.

La risoluzione in energia dello scintillatore si somma quadraticamente alle altre componenti di risoluzione energetica del sistema. I principali fattori che contribuiscono alla risoluzione energetica sono le disomogeneità e non uniformità delle caratteristiche geometriche ed ottiche del cristallo. Si illustrano ora le caratteristiche dei due materiali più utilizzati nella realizzazione di collimatori inorganici facendo presente che possono essere utilizzati anche materiali organici allo stesso scopo. La natura diversa del materiale comporta un differente studio del fenomeno di emissione luminosa che verrà analizzato nel capitolo 2.

Lo ioduro di cesio o lo ioduro di sodio vengono impiegati nella applicazioni dove si ricorre a particolari detector a stato solido (SDD) invece che ai tubi fotomoltiplicatori. Per avere un miglior matching di emissione spettrale con questo tipo di rivelatori si preferisce modificare la composizione del cristallo con l'aggiunta di Tallio Tl ottenendo CsI-Tl oppure NaI-Tl. Il picco di emissione è a 420nm o 550nm rispettivamente. Per l'accoppiamento con i *Semiconductor Drift Detector* SDD i cristalli di CsI-Tl dovrebbero esseree preferiti siccome l'efficienza degli SDD è massima per lunghezza d'onda tra

i 440 e i 700 nm. Tipicamente un gamma camera impiegante PMT utilizza NaI-Tl per il motivo appena illustrato. Anche la resa ottica dei materiali basati sullo ioduro di cesio è buona e cioè intorno ai 42-65 fotoni/KeV ma lo svantaggio risiede nella loro costante di decadimento relativamente lunga dell'ordine dei 0.6-3μs che di fatto costituisce una grossa limitazione sia per la risoluzione energetica del sistema che per la capacità di conteggio degli eventi per una gamma camera impiegante potodetector ad elevate prestazioni. Inoltre anche la risoluzione energetica è piuttosto scadente con valori dal 5,9% al 6,6% per raggi gamma a 662keV.

Bromuro di Lantanio. Alcuni recenti scintillatori inorganici drogati con Cesio ($RbGd_2Br_7 : Ce$, $LaCl_3 : Ce$, $LaBr_3 : Ce$, $LuI_3 : Ce$) possiedono efficienza di scintillazione e risoluzione energetica molto vicina ai limiti teorici così come pure tempo di decadimento veloce e densità moderatamente elevata. Per questi motivi $LaBr_3 : Ce$ e $LaCl_3 : Ce$ hanno la potenzialità di sostituire $NaI : Tl$. Il primo tra i due è sicuramente il più promettente; la sua resa ottica è di 61 fotoni/keV ovvero il 50% in più rispetto al $NaI : Tl$ nonché il valore più alto tra i materiali scintillatori inorganici (la generazione di luce dipende dalla concentrazione dello ione Ce^{3+}). Il picco di emissione dei cristalli in $LaBr_3 : Ce$ è tra 360 e 380nm ovvero molto vicino alla lunghezza d'onda corrispondente al picco massimo di efficienza di un fototubo PMT ed ancora nella regione spettrale significativa per l'impiego di detector a stato solido.

1.5 PMT

La luce generata del cristallo scintillatore deve essere convertita in segnali elettrici per poter essere misurata dall'elettronica successiva al rivelatore. Per questo scopo si utilizzano i tubi fotomoltiplicatori o dei detector a giunzione ad esempio SDD. La struttura e lo schema di funzionamento sono illustrati in figura 1.5.

Tipicamente un fotocatodo mantenuto a potenziale negativo elevato (centinaia di Volts), un elettrodo collettore chiamato anodo e una serie di elettrodi

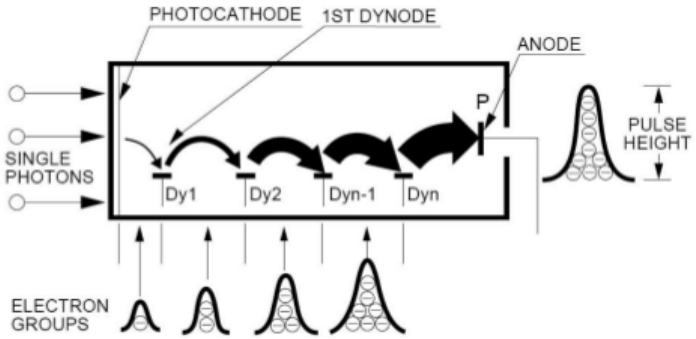

Figura 1.4: *Schema di principio di un detector fotomoltiplicatore.*

ausiliari denominati dinodi sono contenuti all'interno di un'ampolla di vetro entro cui viene applicato il vuoto.

Il fotocatodo è la prima struttura costituente un PMT ed è responsabile della conversione luce carica elettrica che caratterizza un generico fotorivelatore. I fotocatodi possono appartenere a due tipologie diverse; spessi oppure sottili. Nel primo caso l'espulsione dell'elettrone fotogenerato avviene dallo stesso lato di incidenza della radiazione luminosa mentre nel secondo caso il catodo è posizionato in corrispondenza di una finestra illuminata dell'ampolla e gli elettroni vengono emessi dal lato opposto rispetto a questa. Si definisce efficienza quantica η del processo di fotogenerazione il rapporto tra il numero di foto-elettroni emessi rispetto al numero di fotoni incidenti. Valori tipici per questa figura di merito sono 20-30% con variazioni dipendenti dalla lunghezza d'onda incidente. Per questo motivo nella scelta di un fotocatodo è necessario valutare la sua efficienza quantica per la lunghezza d'onda corrispondente al picco massimo di emissione del cristallo scintillatore che si intende utilizzare.

L'energia che può essere trasferita da un fotone ad un elettrone è data da $h\nu$ dove h è la costante di Planck e v la frequenza dell'onda. Parte di questa energia viene persa in seguito agli urti derivanti dalla migrazione dell'elettrone verso la superficie del materiale del catodo e la restante parte, se superiore al lavoro di estrazione, definito come l'energia necessaria a far fuoriuscire un elettrone dal materiale, caratterizza l'energia cinetica dell'elettrone al momento dell'espulsione.

Una possibile fonte di rumore è quella dovuta al rumore termoionico associato all'emissione spontanea di elettroni dal catodo per agitazione termica.

Considerata la polarizzazione del fotocatodo gli elettroni vengono accelerati verso il complesso di dinodi posizionati secondo una geometria opportuna al fine di massimizzare l'efficienza del meccanismo di emissione elettronica secondaria che caratterizza la moltiplicazione degli elettroni nel PMT. I dinodi sono a loro volta polarizzati con tensioni decrescenti fino all'ultimo che viene rivolto verso l'anodo, uscita utile del dispositivo. La moltiplicazione ha luogo in quanto l'energia dell'elettrone accelerato dal campo elettrico tra fotocatodo e primo dinodo è abbastanza elevata da produrre, in seguito alla collisione dell'elettrone con il dinodo stesso, un buon numero di elettroni che accelerati a loro volta dai campi elettrici intermedi ai dinodi produrranno un numero sempre crescente di elettroni. Impiegando N dinodi è possibile ottenere guadagni dell'ordine di 10^5 ma è importante sottolineare che tale meccanismo implica l'introduzione di fonti di rumore dovute alla natura Poissoniana del processo. Generalmente il rumore dominante per un PMT, in virtù dell'elevato guadagno che rende trascurabile il rumore dell'elettronica di preamplificazione, è quello di corrente di buio del fotocatodo che comunque permette di ottenere ottimi valori di rivelazione (anche poche decine di fotoni con filtraggio ottimale).

Nella gamma camera il numero di fotomoltiplicatori impiegati va da una ventina fino ad un centinaio, e vengono raggruppati in un array esagonale come mostrato in figura 1.5.

I segnali elettrici in uscita da tutti i PMT vengono elaborati da appositi circuiti in grado di localizzare le coordinate dell'evento di interazione sullo scintillatore. Una rete di resistori o capacitori, che trasferiscono una quantità di segnale proporzionale alla distanza del PMT dal centro del cristallo, forniscono l'informazione della posizione. Nonostante il numero limitato di sensori e possibile ricostruire un'immagine con risoluzione di 256x256 o 512x512 pixel interlacciando tutti i segnali prodotti da un certo evento. La somma dei segnali in uscita da tutti i PMT produce un impulso di ampiezza direttamente proporzionale all'intensità della luce prodotta nel cristallo; questo fornisce una stima dell'energia totale del fotone. Un circuito di analisi di questo se-

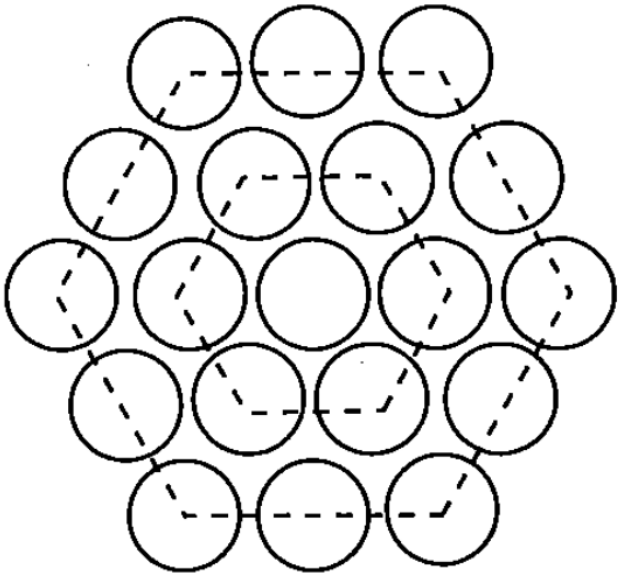

Figura 1.5: *Disposizione esagonale dei detector in una gamma camera.*

gnale permette di rigettare tutti gli eventi che non rientrano in un certo gap di energie prescelto dall' operatore. Si realizza in questo modo un trigger che consente di eliminare gran parte delle radiazioni diffuse per effetto Compton.

1.6 SDD

L'SDD *(Silicon Drift Detector)* ideato da E. Gatti e P. Rehak nel 1983 è un detector di radiazioni ionizzanti caratterizzato da una capacità molto piccola dell'elettrodo che raccoglie la carica dei segnali. Il basso rumore elettronico ottenibile con un SDD, grazie al basso valore di capacità in uscita, realizzato al meglio integrando il transistor di front-end nel chip del detector, ha reso l' SDD un detector ideale per la rivelazione dei raggi X per le misure spettroscopiche ad elevata risoluzione. Dal momento in cui è stato ideato l' SDD è stato sviluppato in molteplici tipologie per applicazioni che spaziano dal campo della fisica delle alte energie a quello della spettroscopia a raggi X.

L' SDD, accoppiato con scintillatori, recentemente si è rivelato essere un dispositivo competitivo nell'ambito della rivelazione di radiazione γ e x ri-

spetto ai fotodetector convenzionali, grazie alla sua elevata efficienza quantica e basso rumore elettronico. Nel seguito si descriverà il comportamento e le performance dell' SDD sottolineandone i vantaggi potenziali nel campo della medicina nucleare rispetto alle alternative commerciali e ai fotodetector allo stato dell'arte.

1.6.1 Principio di funzionamento

Un detector a semiconduttore a deriva consiste in un volume di silicio di tipo n svuotato confinato tra due regioni di tipo p posizionate su entrambi i lati (superiore ed inferiore) della struttura.

Il suo principio di funzionamento può essere compreso iniziando dal concetto illustrato in figura 1.6 dello svuotamento laterale.

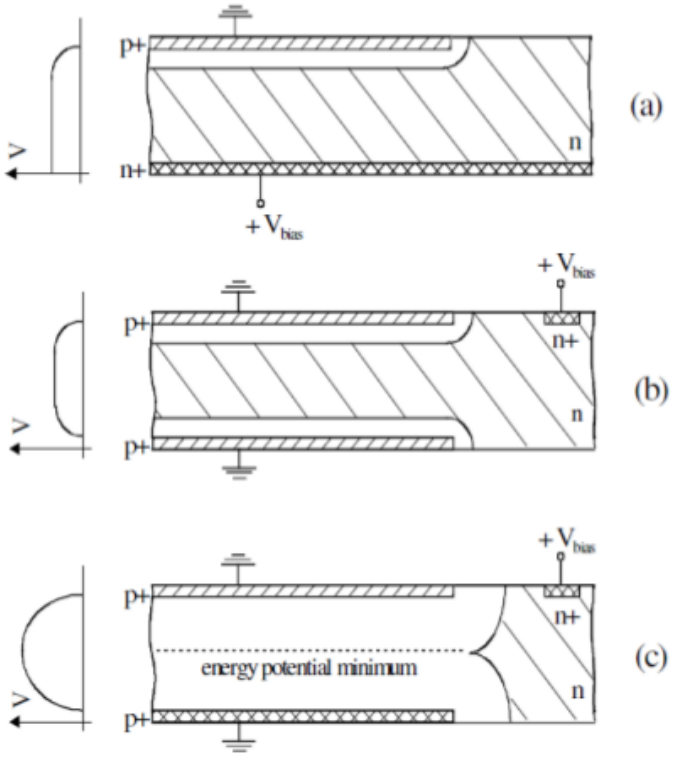

Figura 1.6: *Illustrazione dello svuotamento laterale in un SDD.*

La figura mostra un diodo p-n convenzionale, dove il contatto della regione ohmica n^+si estende sull'intera area del wafer. Lo svuotamento del bulk può essere ottenuto polarizzando positivamente un piccolo elettrodo n^+rispetto agli elettrodi p^+posizionati su entrambi i lati del wafer. Quando la tensione di polarizzazione applicata è abbastanza elevata le due regioni di carica spaziale separate dalla regione di bulk non svuotata vengono in contatto lasciando una piccola regione n non svuotata solo in corrispondenza dell'elettrodo n^+. Lo svuotamento del bulk viene così ottenuto applicando una tensione che è quattro volte inferiore rispetto alla tensione necessaria per un diodo convenzionale con lo stesso spessore.

In accordo con il meccanismo di polarizzazione descritto il diagramma dell'energia potenziale per un elettrone perpendicolare alla superficie del wafer ha un andamento parabolico con un minimo localizzato al centro del wafer stesso.

Un campo elettrico addizionale, parallelo alla superficie del wafer, viene inserito per forzare gli elettroni, nel minimo dell'energia potenziale, a spostarsi verso l'anodo n^+. Il tempo di deriva degli elettroni può essere utilizzato per misurare le coordinate di interazione mentre la carica raccolta consente di misurare l'energia rilasciata dall'evento di interazione ionizzante. Il campo trasversale viene implementato impiantando un array di elettrodi p^+invece di uno singolo. Un partitore di tensione integrato, implementato attraverso p-mosfet ad arricchimento operanti in regione ohmica, viene utilizzato per polarizzare i segmenti p^+. Nello specifico, per applicazioni spettroscopiche, la giunzione p posteriore è una struttura equipotenziale che costituisce la finestra d'ingresso della radiazione. Particolare attenzione è stata dedicata al design di tale finestra limitando il più possibile l'area non sensibile ed ottimizzando il profilo di drogaggio per minimizzare la perdita di carica.

Gli elettroni, una volta generati dalla radiazione ionizzante, scivolano nel minimo del canale di potenziale e vengono trasportati verso l'anodo mentre le lacune, trasportate dal campo di svuotamento, sono velocemente raccolte dai vicini elettrodi p^+. Il minimo del potenziale viene fissato in corrispondenza dell'anodo stesso polarizzando opportunamente gli elettrodi. La nuvola elettronica induce all'anodo un impulso di uscita solo quando essa giunge in

prossimità dello stesso a causa dell'azione schermante degli elettrodi p^+.

Il vantaggio principale di un SDD rispetto ad un diodo pn convenzionale di uguale area attiva e spessore è nel basso valore di capacità dell'anodo collettore, dell'ordine di $100/200\ fF$. Questo valore,inoltre, è indipendente dall'area attiva. Questa caratteristica consente di ridurre sia il rumore dell'elettronica che il valore del tempo di shaping necessario per il processing dei segnali.

1.6.2 L'elettronica on-chip

Per godere a pieno dei benefici, in termini di risoluzione energetica e breve tempo di shaping, derivante dalla bassa capacità di uscita. tipica dell' SDD, sia la capacità di ingresso del preamplificatore che le capacità parassite del collegamento tra detector e preamplificatore devono essere mantenute il più piccole possibile. Questo obiettivo può essere raggiunto attraverso l'integrazione del transistore di front-end del preamplificatore di carica direttamente sul detector. Il transistore è un n-jfet non convenzionale progettato per operare su silicio completamente svuotato ad alta resistività essendo posizionato all'interno dell'anodo sagomato ad anello. Il fet opera in una configurazione a source follower e viene polarizzato attraverso un terminale esterno ed un generatore di corrente.

La rappresentazione schematica di un SDD per spettroscopia a raggi X con il transistore integrato è rappresentata in figura 1.7. In aggiunta a quanto appena illustrato, è da sottolineare che anche tutti i dispositivi per la scarica dei segnali e la compensazione delle correnti di perdita vengono integrati sul chip.

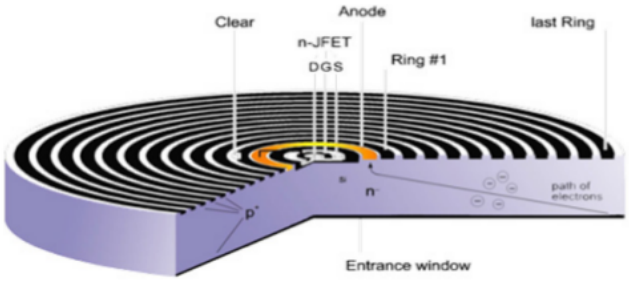

Figura 1.7: *Schema di un SDD con transistore di frontend integrato.*

Capitolo 2

Obiettivi

L'evoluzione dei sistemi di elaborazione digitale per la misura di radiazione ha messo in evidenza la convenienza di sviluppare tecniche sempre più sofisticate per l'emulazione dei sistemi di acquisizione fino a simulare sorgenti radioattive reali. Grandi sono i vantaggi derivanti dall'utilizzo di emulatori a dispetto di vere sorgenti di radiazione nucleare. Prima fra tutte vi è la possibilità per i tecnici delle apparecchiature di rivelazione di lavorare in ambienti a basso rischio, specie in fase di collaudo e messa a punto: l'utilizzo prolungato di un emulatore non ha di fatto controindicazioni diversamente dall'esposizione a radiazione. La drastica riduzione dei tempi e dei costi necessari per effettuare debug dei macchinari di diagnostica nucleare migliora poi la qualità degli esperimenti in virtù della capacità dell'emulatore di costruire segnali arbitrari mirati alla risoluzione del problema. Inoltre tutte le operazioni possono essere effettuate anche da remoto evitando che il personale qualificato debba essere presente in loco.

In ambito biomedicale l'utilizzo di macchine tomografiche ad alta risoluzione permette di visualizzare in dettaglio l'anatomia di parti del corpo del paziente. Queste sfruttano massicciamente tutta la tecnologia messa a disposizione dall'elettronica nucleare. La volontà di poter effettuare imaging biomedico porta all'esigenza di dover effettuare rivelazione matriciale. Ciò può essere ottenuto, come visto nel precedente capitolo, con l'impiego di matrici di sensori, similmente a ciò che viene fatto nell'imaging fotografico. Viene naturale ritenere, allora, che anche l'emulazione nucleare debba approcciar-

si all'imaging bidimensionale in modo tale da poter seguire con efficacia la naturale evoluzione degli apparati di rivelazione verso tale direzione.

Il progetto consiste, pertanto, nella realizzazione di un emulatore multicanale per radiazione nucleare. Si vuole cioè realizzare un sistema elettronico mixed signal la cui peculiarità sia quella di generare forme d'onda analogiche che siano equivocabili con quelle provenienti da reali matrici di detector di radiazione nucleare. In particolare ci si è posti l'obiettivo di realizzare un sistema che potesse simulare una matrice di detector di dimensione arbitraria fino ad un massimo di 8 x 8.

Possiamo dire, in definitiva, che il sistema vuole emulare una Anger camera in tutti gli aspetti che la caratterizzano.

Nella prima parte del capitolo analizzeremo il modo in cui viene effettuata la rivelazione di radiazione gamma attraverso l'utilizzo di una gamma camera partendo dalla generazione dei fotoni stessi fino ad arrivare ai segnali elettrici in uscita.

Basandoci su questa descrizione, nella seconda parte mostreremo lo scopo che ci si è preposti offrendo al lettore una visione di insieme del processo di emulazione.

2.1 Il processo di rivelazione

Diversi sono gli aspetti da considerare per poter generare dei segnali quanto più vicini possibile al caso reale. Una miglior comprensione della fisica del fenomeno di rivelazione permetterà di capire le peculiarità tipiche dell'emulazione.

Consideriamo, dunque, di allestire un apparato sperimentale costituito da una sorgente radioattiva, quale può essere ad esempio un radiofarmaco iniettato in una cavia, una gamma camera e un sistema di misura accoppiato ad essa. Per semplicità ipotizziamo che il collimatore impiegato sia di tipo a setti paralleli così da consentire, come illustrato nel capitolo 1, la rivelazione dei soli fotoni normali al piano della gamma camera. I fotodetector impiegati siano PMT (o SDD) scelti in modo tale da avere un buon accoppiamento ottico con il cristallo scintillatore.

Come sappiamo, quando un fotone gamma incide sullo scintillatore esso interagisce col materiale di cui questo è composto e produce un fascio di fotoni ottici che vengono rivelati dai detector. Per valutare tutte le fasi dell'esperimento in dettaglio possiamo suddividere l'analisi in due parti. Lo spartiacque fra le due è costituito dalla gamma camera, nel senso che, tutti i fenomeni dell'esperimento che avvengono all'esterno della gamma camera appartengono alla prima parte mentre tutti quelli che avvengono all'interno appartengono alla seconda.

Analizziamo dunque dapprima le caratteristiche della radiazione che giunge alla gamma camera e successivamente analizziamo il meccanismo che permette di ottenere dei segnali elettrici dai detector.

2.1.1 Generazione della radiazione

Alla luce delle conoscenze acquisite, oggi sappiamo che nel nucleo di un atomo vi sono Z protoni ed N neutroni e che Z identifica il numero atomico mentre il numero di massa A è dato dalla somma N+Z, cioè del numero di protoni e neutroni presenti nel nucleo di quel elemento. Un nuclide di un elemento è individuato dai valori di A e di Z ed il suo simbolo è: ${}^{A}_{Z}E$. Nuclidi con lo stesso numero atomico (Z) e diverso numero di massa (A) sono detti *isotopi*: ${}^{238}_{92}U$,${}^{235}_{92}U$.

Fino al 1919 i soli fenomeni nucleari conosciuti erano quelli legati alla radioattività naturale prodotta da quelli che erano allora gli ultimi 12 elementi del sistema periodico, con Z da 81 a 92. Oggi invece si conoscono più di 1400 nuclidi. Ogni isotopo radioattivo è caratterizzato da alcuni parametri fisici che ne consentono l'identificazione: il tempo di dimezzamento fisico, cioè il tempo nel quale la radioattività originaria si dimezza, il tipo di emissione radioattiva (alfa, beta, gamma) e l'energia della radiazione emessa.

Le particolarità del Tecnezio ${}^{99m}Tc$, con le sue 6 ore di tempo di dimezzamento ed emissione radioattiva di tipo gamma, ne hanno fatto uno degli isotopi più utilizzati nei radiofarmaco in medicina nucleare. L'emissione radioattiva di tipo gamma avviene in seguito ad una transizione *isomerica* ovvero senza variazione delle proprietà chimico fisiche dell'elemento ma soltanto

energetiche. In particolare si osserva la seguente trasformazione:

$$^{99m}Tc- > ^{99}Tc + \gamma$$

derivante a sua volta dal decadimento beta del ^{99}Mo. E' importante sottolineare che la radiazione emessa in questa particolare situazione ha un'energia di $E_0 = 140KeV$ e che successivi decadimenti sono separati da intervalli temporali di natura statistica, come vedremo in seguito.

2.1.1.1 Statistica energetica

La radiazione, una volta generata, deve attraversare i tessuti del corpo, caratterizzati da atomi a basso numero atomico, ove, tipicamente, subisce interazioni di tipo Compton.

In questo processo (figura 2.1) sono coinvolti la radiazione gamma ed un elettrone di un atomo del tessuto biologico. Quando un fotone gamma incide su un elettrone del tessuto si osserva una brusca variazione della sua traiettoria e una diminuzione della sua energia originaria. La frazione di energia persa viene ceduta all'elettrone e la sua entità dipende dall'angolo di scattering θ secondo l'espressione:

$$h\nu^{'} = \frac{h\nu}{1 + \frac{h\nu}{mc^2}(1 - cos(\theta))}$$

dove mc^2 è l'energia della massa dell'elettrone a riposo.

Per piccoli scostamenti angolari viene trasferita una piccola porzione di energia. Il massimo trasferimento energetico si ha, ovviamente, quando il fotone incidente è *retro scatterato* dall'elettrone (con θ_{max}) e la sua traiettoria originale è invertita. Poiché, in teoria, tutti gli angoli di scattering possono presentarsi, i valori di energia $h\nu'$ vengono prodotti con continuità da un minimo per $\theta = 0$ ad un massimo che si ha per $\theta = \theta_{max}$. Il massimo di energia può essere predetto attraverso la legge di conservazione del momento angolare e la legge di conservazione dell'energia. In ogni caso si comprende che le energie risultanti da tutte le interazioni che un fotone gamma può subire

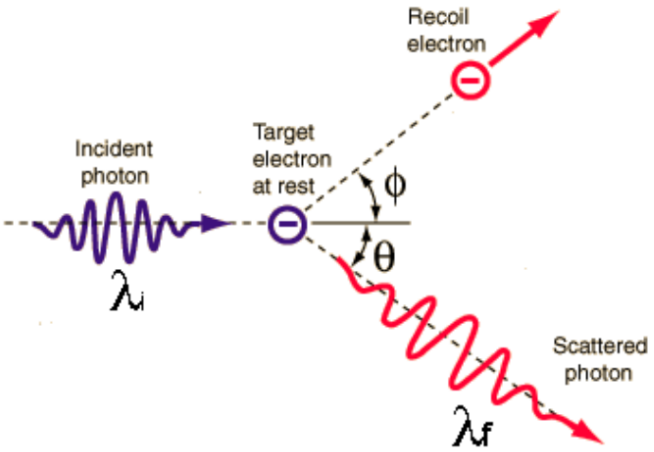

Figura 2.1: Scattering Compton.

devono essere rappresentate secondo una statistica più o meno complessa che ad esempio può avere l'aspetto di quella in figura2.2.

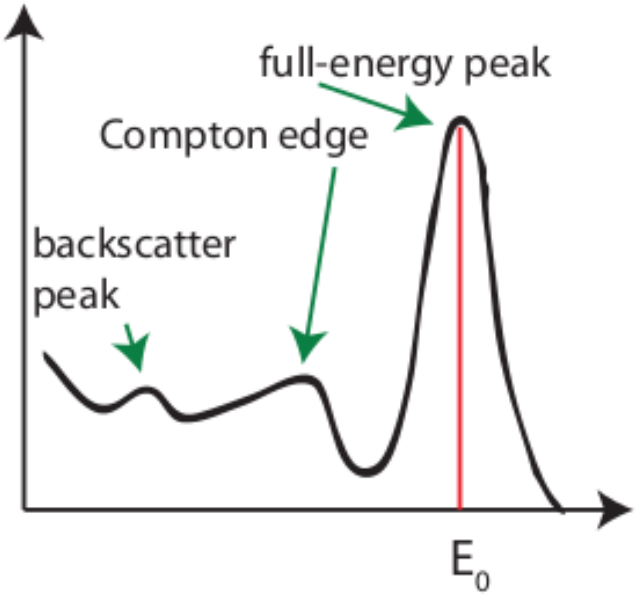

Figura 2.2: *Esempio di uno spettro energetico di emissione.*

2.1.1.2 Statistica temporale

Il decadimento nucleare (ad esempio del Tecnezio) è a tutti gli effetti un processo aleatorio. L'emissione di radiazione non è nient'altro che un rilascio di energia di un sistema che evolve da uno stato ad un altro energeticamente più stabile. In accordo con la fisica classica lo scambio di energia avviene in uno spazio continuo; ogni quantità di energia, pur piccola che sia, può essere scambiata purché sia rispettata la legge di conservazione. L'evoluzione di un sistema è esattamente determinata dalle condizioni iniziali e dalle forze che agiscono su di esso.

La teoria quantistica cambiò radicalmente questa concezione. In accordo con questa l'energia può essere scambiata solo in quanti ovvero in pacchetti di limitata entità. Il rispetto della legge di conservazione dell'energia è, dunque, una condizione necessaria ma non sufficiente affinché avvenga l'evoluzione di un sistema. Il destino di un sistema non è determinato esattamente dalle condizioni iniziali e dalle forze applicate. E' possibile solo parlare di probabilità che il sistema evolva o meno.

Quindi con l'introduzione della teoria quantistica lo studio dei fenomeni fisici cambia da "deterministico" a "probabilistico".

L'emissione di radiazione obbedisce alle leggi della teoria quantistica. Se si cerca di misurare il numero di particelle emesse da una reazione nucleare, si vedrà che non è costante nel tempo. Esso ha un'incertezza statistica a causa della natura probabilistica di questo fenomeno.

Ipotizziamo che si cerchi di misurare il numero di fotoni per unità di tempo, emessi da una certa sorgente radioattiva. Per ogni atomo della sorgente c'è la probabilità, ma non la certezza, che un fotone sarà emesso all'interno della prossima unità di tempo. Un parametro che ci fornisce l'evoluzione del fenomeno è il numero medio di particelle emesse.

Il decadimento radioattivo è un processo stocastico dipendente dal numero di atomi che possono decadere e da una funzione probabilistica che è caratteristica del loro tempo di vita. La probabilità di rivelare uno specifico numero di eventi in una data misura è data dalla distribuzione gaussiana. Tuttavia nella misura di un piccolo numero di eventi la distribuzione di probabilità è differente ed è data dalla distribuzione di Poisson. Per esempio in

un piccolo periodo di tempo il numero di particelle rivelate possono essere in media 5. Una variazione di 1 o 2 eventi è, dunque, significativa rispetto alla media.

La distribuzione di Poisson è un caso particolare della distribuzione binomiale, simile alla distribuzione gaussiana, e soddisfa la seguente relazione:

$$P(n) = \frac{\overline{n}^n}{n!} e^{-\overline{n}}$$

.

Dove $P(n)$ è la probabilità normalizzata che in un dato intervallo di tempo n eventi possano essere osservati, $\overline{n}$ è il numero medio di eventi quando vengono considerati tanti campioni. Con n elevato il suo fattoriale esplode e l' utilizzo della distribuzione Poissoniana non risulta più agevole; in questa circostanza la distribuzione gaussiana è preferibile:

$$P(n) = \frac{1}{\alpha\sqrt{2\pi}} e^{(-\frac{(n-\overline{n})^2}{2\sigma^2})}$$

$\overline{n}$ è il valor medio su cui viene centrata la distribuzione e σ è la deviazione standard dal valore medio.

Per un detector di radiazione con efficienza unitaria che conta l'emissione di un singolo radioisotopo si ha:

$$r = |\frac{dN}{dt}| = \lambda N$$

$$N = m\frac{N_A}{A}$$

dove N è il numero di nuclei radioattivi, m è la massa dell'isotopo, N_A è il numero di Avogadro, A è il peso atomico, λ è la costante di decadimento ed r è il tasso medio di occorrenza degli eventi. E' possibile dimostrare che la funzione di distribuzione che descrive l'intervallo di tempo intercorrente tra due eventi consecutivi ha l'espressione

$$I(t)dt = re^{-rt}dt$$

2.1.2 Interazioni nella gamma camera

Analizziamo ora la seconda parte dell'esperimento, ovvero il susseguirsi di eventi che avvengono a partire dall'interazione con il collimatore.

La radiazione emessa dalla cavia giunge al collimatore con le caratteristiche energetiche e temporali illustrate. A differenza dei tessuti organici il materiale del collimatore rende maggiormente probabile un altro tipo di interazione: l'assorbimento fotoelettrico. In questo processo il fotone gamma incidente scompare completamente e la sua energia viene trasferita ad un elettrone dell'atomo del collimatore. Siccome questa energia eccede di gran lunga quella di legame dell'elettrone, ne consegue che esso viene espulso ad alta velocità. L'energia cinetica di questo elettrone secondario corrisponde a quella del fotone gamma incidente diminuita dell'energia di legame. Il processo porta alla formazione di una lacuna che sarà rapidamente compensata dall'arrivo di un elettrone libero limitrofo. A quest'ultimo processo segue l'emissione di energia sotto forma di radiazione X che conseguentemente interagirà con elettroni debolmente legati ovvero appartenenti ad orbitali esterni. Il risultato finale è la comparsa di elettroni veloci la cui energia è dissipata in seguito ad urti reticolari.

I fotoni che possiedono le caratteristiche per oltrepassare il collimatore giungono sul cristallo scintillatore.

Alla penetrazione all'interno del cristallo della radiazione segue la generazione di fotoni; processo che scaturisce, per gli scintillatori organici, da transizioni energetiche nella struttura a bande della singola molecola. Alcune molecole organiche possiedono la così detta struttura a orbitale π cui corrisponde lo schema di livelli energetici rappresentato in figura 2.3. Si osserva che tutte le molecole a temperatura ambiente sono nello stato S0. L'assorbimento di energia cinetica di una particella carica libera, formatasi per effetto fotoelettrico, può far evolvere il sistema verso uno stato eccitato come ad esempio S1 o S2. Tipicamente lo stato S1 è quello più probabile di metastabilità. La principale fluorescenza viene emessa in seguito alla transizione dallo stato S1 allo stato S0 ed esibisce un tipico decadimento esponenziale con costante di tempo di pochi ns. L'andamento dell'impulso di luce è caratterizzato da una componente veloce dominante ed una lenta. Quella veloce è

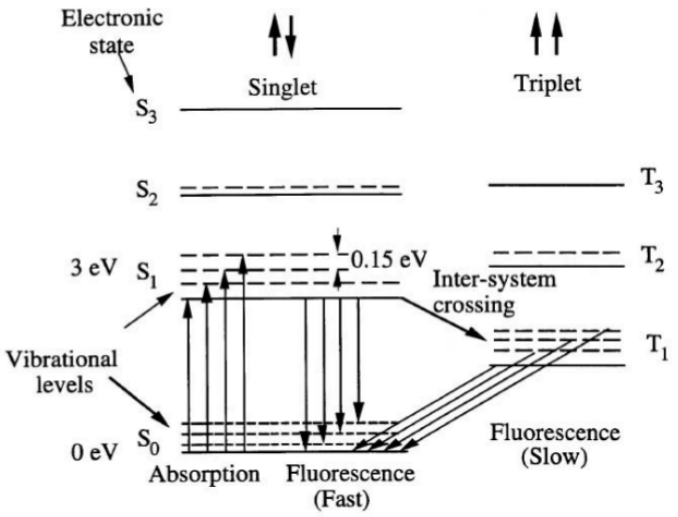

Figura 2.3: *Configurazione degli orbitali elettronici di un cristallo scintillatore organico.*

la fluorescenza appena spiegata mentre quella lenta è una fluorescenza ritardata che può perdurare per alcune centinaia di nanosecondi. La coda lenta del bagliore dipende dalla natura della particella ionizzante il che può essere utilizzato per discriminare il tipo di radiazione.

Per gli scintillatori inorganici, come quelli descritti nel capitolo 1, il discorso è lievemente diverso. In questo caso gli stati energetici sono quelli tipici del reticolo.

Un cristallo puro è caratterizzato da una banda di valenza ed una di conduzione separate da un intervallo proibito di energie denominato *energy gap* E_g. L'assorbimento di energia in questa struttura deriva dalla promozione alla banda di conduzione di un elettrone appartenente inizialmente alla banda di valenza. Il successivo ritorno dell'elettrone eccitato alla banda di valenza, fenomeno che accade in virtù della migliore stabilità energetica del sistema, produce un fotone ad elevata energia pari ad E_g. Essendo quest'ultimo ben lontano dalla regione spettrale del visibile si ha la necessità di ridurre l'energy gap per ottenere uno scintillatore dalle buone proprietà di conversione. A tal scopo si aggiungono al cristallo delle impurità denominate attivatori che alterano come in figura 2.4 la struttura a bande del sistema. Una importante conseguenza nell'utilizzo di tali impurità è che il cristallo risulta trasparente

alla luce di scintillazione siccome la radiazione derivante ha energia minore di quella necessaria alla generazione di una coppia elettrone lacuna (E_g).

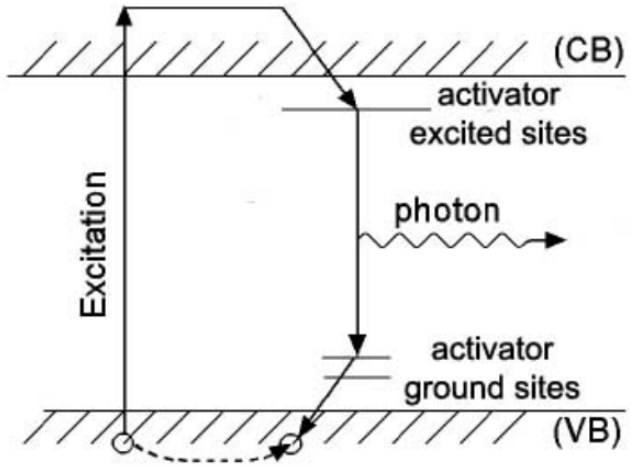

Figura 2.4: *Diagramma a bande di un cristallo scintillatore inorganico.*

Sia per gli scintillatori organici che inorganici, da un punto di vista concettuale, possiamo immaginare che questi siano dei blocchi funzionali con un ingresso matriciale ed una uscita matriciale. L'ingresso del sistema specifica le caratteristiche della radiazione gamma cioè la sua energia e la sua posizione mentre l'uscita fornisce le specifiche della radiazione ottica prodotta. La distribuzione di fotoni in uscita segue un andamento gaussiano bidimensionale come mostrato in figura 3.7. Si comprende allora che un singolo fotone gamma produce sempre l'eccitamento di un buon numero di rivelatori i quali produrranno segnali elettrici coerentemente alla quantità di luce ricevuta. Per questo motivo la lettura dei segnali consente di ricostruire le coordinate di interazione sullo scintillatore e indirettamente il punto di generazione del fotone gamma stesso all'interno del tessuto biologico.

2.1.2.1 Generazione del segnale di uscita

Nel capitolo precedente si è visto che una gamma camera può essere costituita da un numero modesto di fotodetector PMT o SDD. Ogni detector è collegato ad un'elettronica di front-end necessaria per trasportare i segnali elettrici

prodotti all'esterno dell'apparato. Lo schema circuitale tipico impiegato è presentato in figura 2.5.

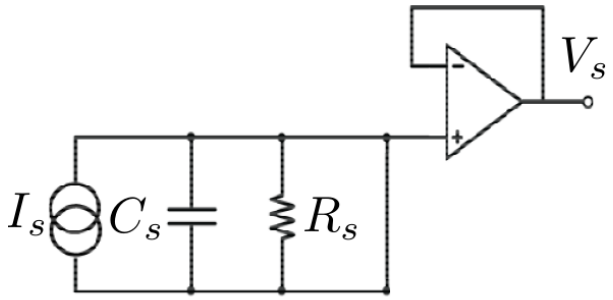

Figura 2.5: *Circuito equivalente per un singolo canale.*

Notiamo che la generazione di fotocorrente del detector viene modellizzata attraverso il generatore di corrente I_s. La corrente di segnale prodotta viene filtrata attraverso la resistenza R_s e la capacità C_s e convertita in un segnale di tensione. La capacità $C_s$include la capacità interna del detector, la capacità esterna dovuta alla realizzazione del sistema e quella in ingresso all'amplificatore; in altre parole è la capacità totale afferente al nodo di ingresso. Il preamplificatore comportandosi da buffer propone in uscita la tensione presente al suo ingresso.

In questa sezione ci proponiamo, dunque, di studiare la forma d'onda tipica in uscita dallo stadio appena illustrato che prende il nome di *canale.*

Un evento di scintillazione gamma produce un numero di fotoni variabile nel tempo in accordo con l'andamento esponenziale della luminescenza accennato poc'anzi. Per la presenza di bagliori di fluorescenza veloci e lenti sarebbe opportuno modellizzare l'andamento del numero di fotoni emessi nel tempo $N(t)$ con la relazione:

$$N(t) = Ae^{-\frac{t}{T_{fast}}} + Be^{-\frac{t}{T_{slow}}}$$

Tipicamente le costanti di tempo dei due esponenziali differiscono per alcuni ordini di grandezza e dipendono dal tipo di materiale componente il cristallo scintillatore. Ad esempio nel caso di scintillatore realizzato in NaI drogato con Tallio si misurano $T_{fast} = 250ns$ e $T_{slow} = 100ms$. Le costanti

A e B rendono il peso degli esponenziali molto diverso nella composizione del bagliore. Sebbene entrambe dipendano dall'energia del fotone, l'esponenziale lenta ha, tipicamente, un peso molto minore rispetto a quella veloce. L'interesse nella rivelazione dell'interazione gamma si concentra sullo studio della componente veloce poiché in prima approssimazione si ha:

$$N(t) \sim \frac{N_0}{T_d} e^{-\frac{t}{T_d}}$$

Dove la costante di tempo T_dè molto vicina a T_{fast}.

Il fotodetector effettua una conversione lineare luce in ingresso - corrente in uscita che viene espressa attraverso la relazione $I_s = P_{in}^{ottica} S_d = N(t) E_{ph} S_d$ dove E_{ph} è l'energia dei fotoni ottici; essendo la sensitività $S_d = \frac{\eta}{E_{ph}}$ otteniamo semplicemente $I_s(t) = N(t)\eta$.

Per questo motivo l'impulso di corrente che viene iniettato sul filtro del *canale* ha l'andamento:

$$i(t) = i_0 e^{-\frac{t}{T_d}}$$

Essendo la carica totale generata da un fotone gamma per un certo *canale* $Q^{tot} = \int i(t)dt = i_0 T_d$ ne deriva che l'altezza dell'impulso di corrente nell'origine è proporzionale all'energia del fotone. Infatti con semplici passaggi risulta $i_0 = \frac{Q^{tot}}{T_d} = \frac{N_0 \eta}{T_d}$ dove il numero di fotoni N_0dipende appunto dall'energia del gamma.

Si ricorda che per calcolare il segnale in uscita da un generico filtro lineare a parametri costanti basta effettuare l'operazione di convoluzione del segnale applicato in ingresso con la risposta all'impulso del filtro $h(t)$. Nel caso del filtro in figura 2.5 risulta semplicemente:

$$h(t) = \frac{1}{T_s} e^{-\frac{t}{T_s}} = \frac{1}{R_s C_s} e^{-\frac{t}{R_s C_s}}$$

Risolvendo per parti l'integrale di convoluzione:

$$V_s(t) = \int_{-\infty}^{\infty} i(\tau) h(t-\tau) d\tau$$

si ottiene il segnale:

$$V_s(t) = \frac{RQ^{tot}}{T_s - T_d}(e^{-\frac{t}{T_s}} - e^{-\frac{t}{T_d}})$$

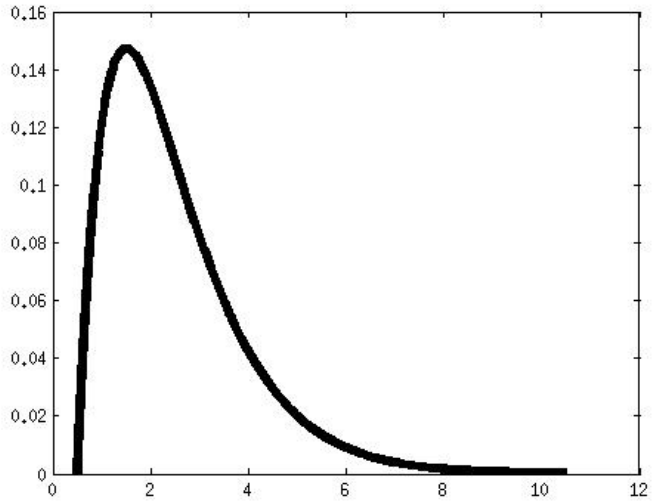

Figura 2.6: *Rappresentazione di* $V_s(t)$.

Lo studio del segnale in uscita dal singolo canale, appena visto, non è sufficiente a caratterizzare completamente il sistema. Come in qualsiasi sistema elettronico anche nella semplice catena di preamplificazione 2.5 sono inevitabilmente presenti sorgenti di rumore.

Al fine di massimizzare il rapporto segnale rumore $\frac{S}{N}$ è indispensabile che i componenti elettronici impiegati nei primi stadi della rivelazione amplifichino il segnale apportando il minimo contributo di rumore. In figura 2.7 vengono rappresentate le sorgenti equivalenti di rumore per il singolo canale.

Il rumore totale in uscita dovrà tener conto dei seguenti contributi:

- Rumore shot dovuto alla polarizzazione del detector
- Sorgenti di rumore interne all'amplificatore
- Rumore Johnson dovuto al resistore equivalente R_s

Al detector è associato un generatore equivalente di rumore shot dovuto sia alla sua corrente di buio I_b che alla corrente di segnale I_s. Essendo però $I_b \gg I_s$ si ha semplicemente $S_i^{detector} = 2qI_b$ con $q = 1.6 * 10^{-19}C$ la

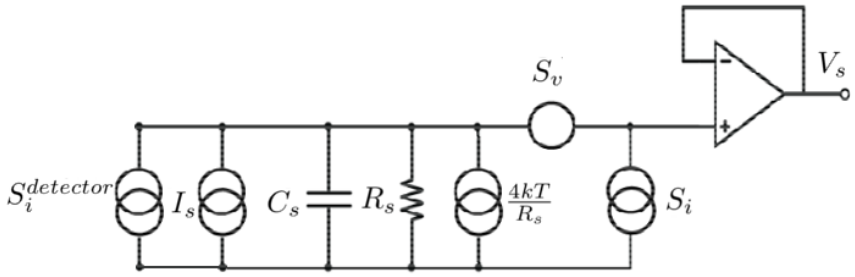

Figura 2.7: *Circuito equivalente di rumore.*

carica elettronica. La resistenza di polarizzazione del detector R_s contribuisce al rumore con un generatore equivalente di valore $\frac{4kT}{R_s}$. L'amplificatore contribuisce sia con un generatore equivalente serie S_v che parallelo S_i, dipendenti chiaramente dal dispositivo, i cui pesi vengono modificati dal valore di R_s. Per bassa impedenza d'ingresso il rumore in tensione dell'amplificatore, riferito al suo morsetto di ingresso, risulta dominante; tipicamente si ha $S_v(f) = S_v^* + \frac{A}{f}$. Il secondo termine dell'espressione rappresenta il noto rumore $\frac{1}{f}$ ed il primo rappresenta la componente bianca.

Tutti i contributi elencati essendo densità spettrali vengono trasferite in uscita attraverso l'espressione:

$$\overline{V}^2_{n,out}(f) = \int_0^\infty (S_v + S_i^{detector} * |Z_s|^2 + \frac{4kT}{R_s}|Z_s|^2)df$$

Dove Z_s è l'impedenza esibita dal filtro.

2.2 Introduzione all'emulazione

La gamma camera coinvolta nell'esperimento di misura appena visto produce dei segnali di tensione con una specifica forma d'onda esponenziale la cui altezza dipende dall'energia rilasciata dal fotone gamma incidente. Utilizzando un sistema di rivelazione nucleare basato su gamma camera è possibile ottenere una rappresentazione visiva, attraverso un'opportuna immagine,

dell'insieme di interazioni gamma rivelate dal sistema. E' solo attraverso questa immagine che può essere fatta la diagnostica medica vera e propria. Per questo motivo la gamma camera viene collegata ad un sistema di calcolo in grado di elaborare ed interpretare nel modo corretto ed in tempo reale i segnali generati e fornire alla fine un immagine che includa tutte le interazioni registrate. Le due operazioni fondamentali necessarie per una corretta costruzione dell'immagine includono la discriminazione dei fotoni scatterati da quelli non scatterati, poiché i primi costituiscono solo rumore nell'immagine finale, e la ricostruzione della posizione di scintillazione. Queste due operazioni vengono realizzate osservando il livello dei segnali prodotti. Supponiamo per fissare le idee che un certo fotone gamma scatterato illumini soltanto 4 detector. Il sistema di calcolo è sensibile alla generazione di segnale è discrimina se il fotone è scatterato oppure no, visto che solo nel secondo caso esso potrà contribuire alla costruzione dell'immagine finale, dopo di che ricostruisce attraverso opportuni algoritmi di media l'esatta coordinata di incidenza. Alla fine dell'esperimento l'insieme dei fotoni non scatterati che il sistema di calcolo è riuscito a discriminare forma l'immagine. Possiamo affermare che quest'ultima fotografa soltanto una situazione di regime. In altre parole non si può dedurre da questa l'esatta successione dei fotoni incidenti nè il tasso temporale di incidenza , nè la presenza di interazioni simultanee. Cioè non interessa; se si vuole osservare la posizione di un tumore nel corpo non importa che alcuni fotoni provenienti dalla periferia della massa siano arrivati prima di altri generati più internamente. Diciamo che l'immagine prodotta in questa situazione deriva da un procedimento canonico di rivelazione. In altre parole allestendo il setup sperimentale proposto nella prima parte si realizza fisicamente la misura.

L'emulazione è una tecnica che consente di simulare il modo di operare di un certo sistema. Si cerca cioè di eguagliare il risultato che si otterrebbe utilizzando il sistema "vero", sfruttando invece l'*emulatore.* Nel nostro caso con emulazione si intende la realizzazione della misura in essenza del setup sperimentale proposto sopra ed in presenza dell'emulatore multicanale. Questo significa che collegando soltanto l'emulatore multicanale al sistema di calcolo è possibile ottenere, attraverso un procedimento non canonico, un'immagi-

ne simile a quella dell'esperimento. Da un certo punto di vista possiamo dire che l'utilizzatore del setup sperimentale "vero" gamma camera è il sistema calcolatore, che invece, in emulazione, viene collegato al setup "virtuale" emulatore.

Per questo motivo è evidente che il sistema di calcolo ricostruttore dell'immagine debba avere l'illusione di essere interfacciato con una vera gamma camera e che i segnali prodotti corrispondano pertanto a vere interazioni gamma.

Nel seguito vedremo come riesce l'emulatore a raggiungere tale obiettivo. L'emulazione multicanale ricorre ad un'immagine di partenza per ottenere, alla fine della rivelazione, la stessa immagine ricostruita dal sistema di calcolo. I passaggi che consentono di ottenere questo risultato sono vari ed articolati ma possiamo sintetizzarli nelle seguenti quattro operazioni:

- **Emulazione posizione di incidenza**
- **Emulazione statistica energetica**
- **Emulazione statistica di emissione temporale**
- **Ricostruzione del segnale (Emulazione forma dell'impulso)**

Riferiamoci alla gamma camera dell'esperimento che deve essere emulata per spiegare il ruolo di queste operazioni.

Sappiamo che l'immagine CT rappresenta la sovrapposizione degli eventi di incidenza. Poiché questi si verificano in successione e producono una sequenza di segnali, di primo acchito si potrebbe pensare di dover proporre al sistema di calcolo ricostruttivo la stessa successione degli eventi. In effetti più che riproporre la stessa esatta sequenza di eventi quello che realmente interessa è ottenere una uguale distribuzione degli eventi. Infatti, visto che l'immagine viene osservata dopo un tempo di misura T, la permutazione tra più eventi nel tempo è del tutto trasparente ai fini della ricostruzione dell'immagine. Banalmente se si colora una pagina e se ne osserva il risultato a lavoro ultimato non si può sapere da quale punto si è iniziato a colorare e non cambia il risultato finale. Il meccanismo che permette la simulazione

dell'ordine e della posizione di arrivo dei fotoni gamma sulla gamma camera è l'emulazione della posizione di incidenza.

Parlando di emulazione di statistica energetica si intende, invece, il fatto che ad ogni fotone simulato è necessario associare un certo livello energetico coerente con la circostanza che in un esperimento reale i fotoni subiscono interazioni Compton. A riguardo è necessaria una precisazione poiché il sistema di misura è in grado di riconoscere fotoni Compton scatterati da fotoni primari non scatterati diversamente da quanto visto per l'ordine di arrivo. Generando soli fotoni non scatterati l'immagine verrebbe ricostruita senza problemi, tuttavia l'emulatore, in questo caso, non risponderebbe con esattezza alle specifiche richieste. In altre parole il sistema di calcolo ricostruttivo si "accorge" che al suo ingresso non è presente una reale gamma camera poiché tutti gli eventi prodotti sono validi per la ricostruzione.

L'emulazione della statistica temporale è un meccanismo che permette di separare l'arrivo di due fotoni simulati con un certo intervallo di tempo variabile in accordo con la specifica statistica di emissione.

L'ultimo step della catena di emulazione è costituito dalla ricostruzione del segnale. Questo passaggio oltre ad essere importante per l'interfacciamento elettrico permette la costruzione di forme d'onda uguali a quelle in uscita da una gamma camera. Poiché l' emulatore si interfaccia al sistema di calcolo ricostruttore solo attraverso i segnali di uscita, questi devono contenere tutte le informazioni derivanti dalle tre fasi dell'emulazione elencate.

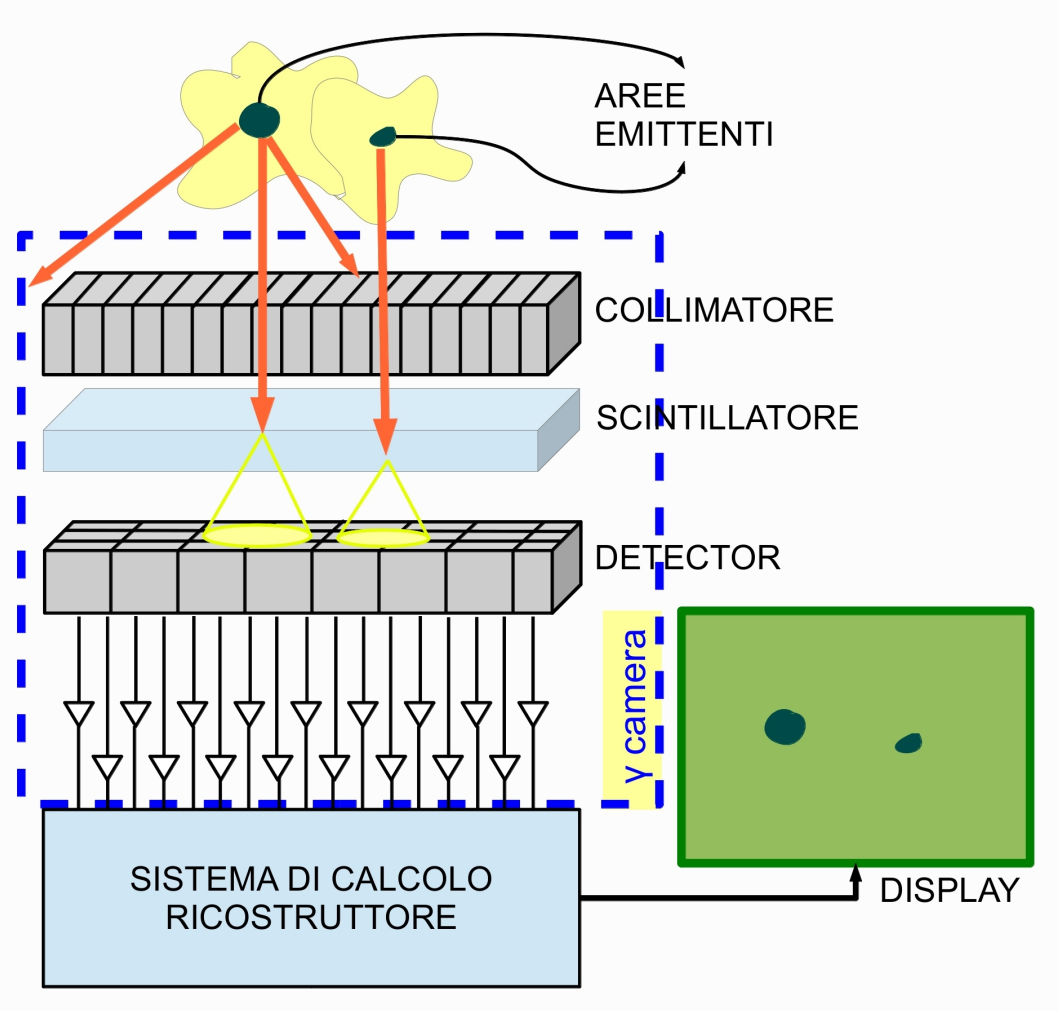

Figura 2.8: *Rappresentazione del processo di rivelazione. Le aree emittenti radiazione gamma vengono visualizzate come dei punti scuri sul display.*

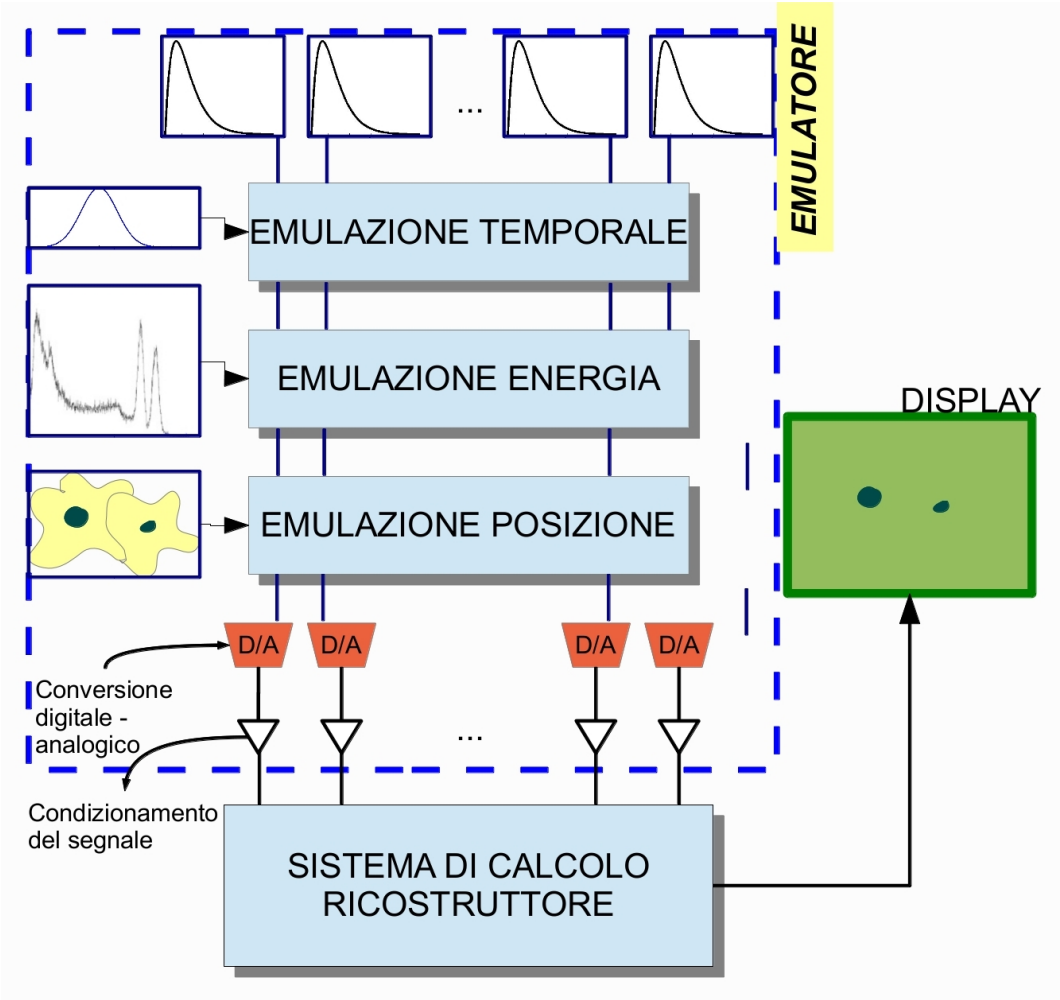

Figura 2.9: *Rappresentazione del processo di emulazione. I punti scuri sul display non rappresentano vere aree emittenti ma l'immagine di test.*

Capitolo 3

Flusso dell'emulazione

L'obiettivo che ci si è preposti è quello di realizzare un emulatore configurabile di gamma camera, ovvero un dispositivo in grado di sostituire una gamma camera reale all'interno del sistema di imaging. In questo capitolo vedremo come l'emulazione viene realizzata. Il processo che permette di costruire dei segnali analogici coerenti con l'esperimento di rivelazione illustrato nel precedente capitolo è un'elaborazione numerica in tempo reale che si compone di più fasi. Al termine di ognuna si ha aggiunta di contenuto informativo.

Per capire come il sistema funzioni possiamo immaginare di osservare una catena di montaggio all'opera. Il prodotto realizzato nella catena di montaggio "emulatore" è dato dall'insieme dei segnali emulati, e si costruisce passo passo sulla base delle diverse statistiche associate alla radiazione gamma attraverso le varie fasi dell'emulazione. Ognuna di queste viene finalizzata all'interno di strutture hardware/software specializzate che, modificando i dati "grezzi" ricevuti in ingresso, forniscono in uscita dati più complessi.

In figura 3.1 possiamo vedere lo schema di principio che realizza l'emulazione.

Ogni percorso verticale si riferisce ad un canale di emulazione della gamma camera ovvero ad un certo detector, con la relativa preamplificazione, che deve essere emulato. L'interfaccia di uscita dell'emulatore, cioè quella che viene fisicamente collegata al sistema di ricostruzione, è costituita da 64 uscite analogiche realizzate attraverso 64 stadi di conversione digitale analogico. Nello schema di figura 3.1 possiamo distinguere chiaramente sulla sinistra

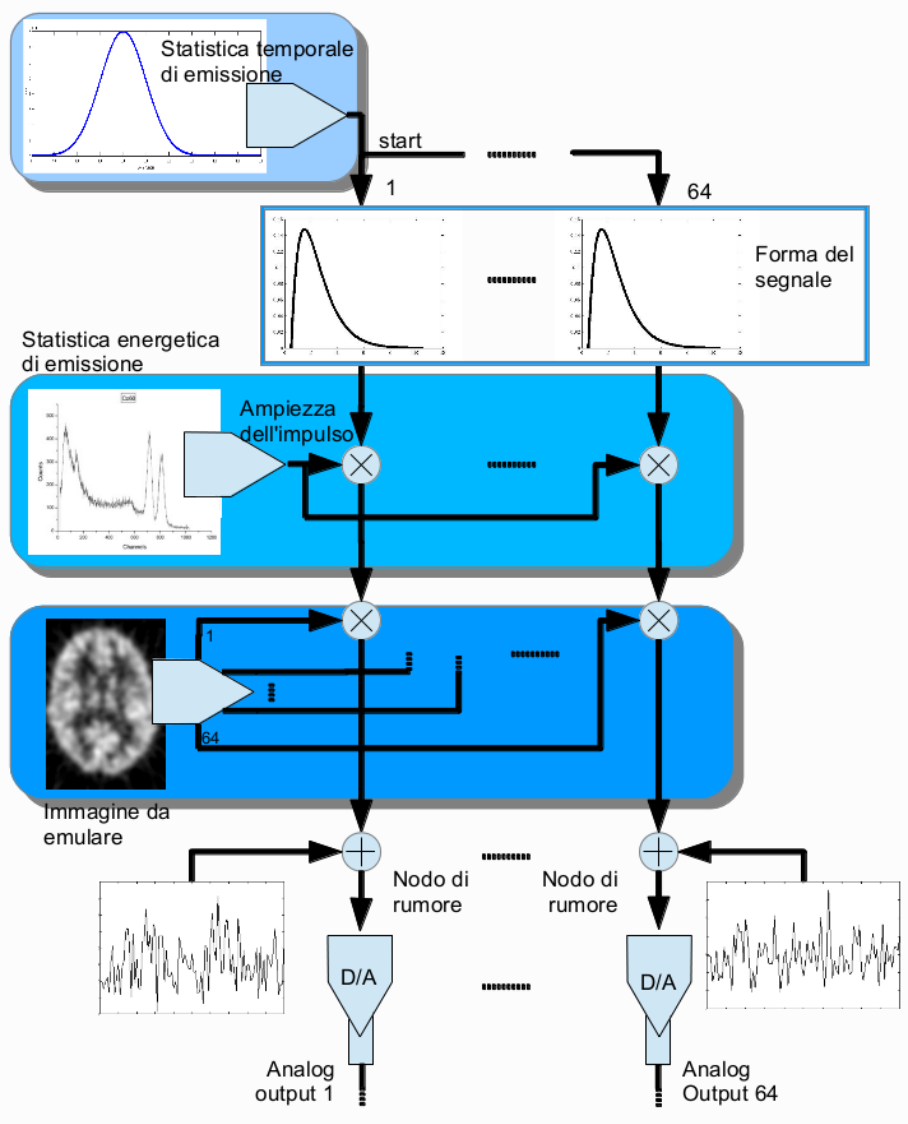

Figura 3.1: *Architettura dell'emulazione.*

i 3 moduli di emulazione fondamentali ovvero l'emulazione della statistica temporale, l'emulazione della statistica di emissione energetica e l'emulazione della posizione di incidenza.

Illustriamo brevemente il funzionamento prima di addentrarci nella spiegazione dei vari blocchi. Quando viene lanciato un segnale di start da parte del modulo emulazione temporale per tutti i canali viene stimolata la trasmissione della forma d'onda standard lungo il percorso di elaborazione. L'altezza

degli impulsi viene scalata in accordo con la statistica energetica di emissione della radiazione attraverso i nodi moltiplicatori superiori. In uscita a questi si ottengono degli impulsi uguali per tutti i canali. L'informazione relativa alla posizione di incidenza viene aggiunta attraverso i successivi moltiplicatori. Il blocco che realizza questa emulazione ha 64 uscite collegate ognuna ad un moltiplicatore; in questo modo si differenziano i segnali dei canali modificando selettivamente l'altezza dei vari impulsi. Infine alle forme d'onda così ottenute si sovrappone il tipico rumore che caratterizza i segnali della gamma camera prima che queste vengano effettivamente convertite in segnali analogici.

3.1 Emulazione spettro di energia

L'emulazione dello spettro di energia è un meccanismo che produce una successione di valori con distribuzione coincidente con una di riferimento. Questi valori vengono utilizzati per modificare l'altezza degli impulsi nella catena di emulazione. La distribuzione di riferimento è quella delle energie della radiazione gamma che si vuole emulare e viene programmata all'interno del sistema. E' possibile derivare in diversi modi questa distribuzione di energie. Una misura reale è sicuramente il migliore modo per ottenere uno spettro fedele, tuttavia ciò ha l'inconveniente di dover utilizzare un reale sistema di rivelazione. Nella pratica si può sfruttare una conversione di una immagine rappresentante lo spettro di interesse nello spettro vero e proprio.

Al fine di spiegare il funzionamento di questa emulazione prendiamo in considerazione uno spettro di energie rappresentato attraverso un istogramma di riferimento. Siccome le energie dei fotoni gamma soddisfano una certa distribuzione di probabilità è intuitivo osservare che alla base della sintesi dello spettro vi sia la generazione di numeri casuali (*RNG, Random Number Generation*). Infatti, qualsiasi variabile statistica x a cui è associata una funzione densità di probabilità $F(x)$, può essere modellizzata attraverso la cascata di un generatore di numeri casuali uniformemente distribuiti e una funzione di trasferimento $F(x)$ come illustrato in figura3.2.

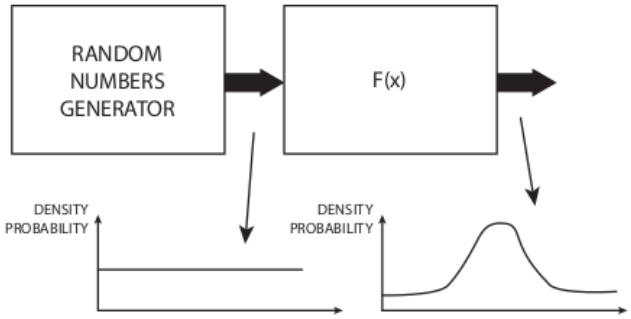

Figura 3.2: *Generazione di numeri che rispettano una statistica F(x).*

Con questo approccio la bontà dei valori statistici sintetizzati dipende esclusivamente dal generatore di numeri casuali, che può essere utilizzato per qualsiasi sorgente da emulare caratterizzata esclusivamente dalla sua funzione $F(x)$. E' pertanto necessario trasformare uno spettro bianco in un altro qualsiasi.

Per spiegare come funziona l'algoritmo scegliamo di utilizzare un approccio a valori discreti. Rappresentiamo una generica funzione densità di probabilità attraverso un istogramma. Per fissare le idee questo istogramma può rappresentare la distribuzione energetica dei fotoni gamma scatterati. Scegliamo, dunque, arbitrariamente che l'istogramma dello spettro sia composto da 16 campioni di energia da E_0 a E_{15} con un range massimo di 16. La larghezza del campione rappresenta la risoluzione spettrale mentre il range è la massima altezza delle colonne dell'istogramma. E' evidente che maggiore è il numero di campioni ed il range massimo , migliore è lo spettro rappresentato. Possiamo pensare di suddividere ciascuna colonna dell'istogramma in un certo numero di piccoli quadrati; se un generico campione x è alto il doppio rispetto ad un altro campione y allora la probabilità per un evento di avere energia E_xè doppia rispetto all'energia E_y. Infatti l'altezza della colonna associata al campione x rappresenta la densità di probabilità che l'evento abbia energia compresa tra E_{x-1} e E_{x+1}.Il prodotto del valore della colonna per l'ampiezza del campione restituisce la probabilità. Il rapporto delle probabilità che un evento abbia energia in un certo intervallo piuttosto

che in un altro è semplicemente il rapporto tra le corrispondenti aree al di sotto della cura di densità di probabilità.

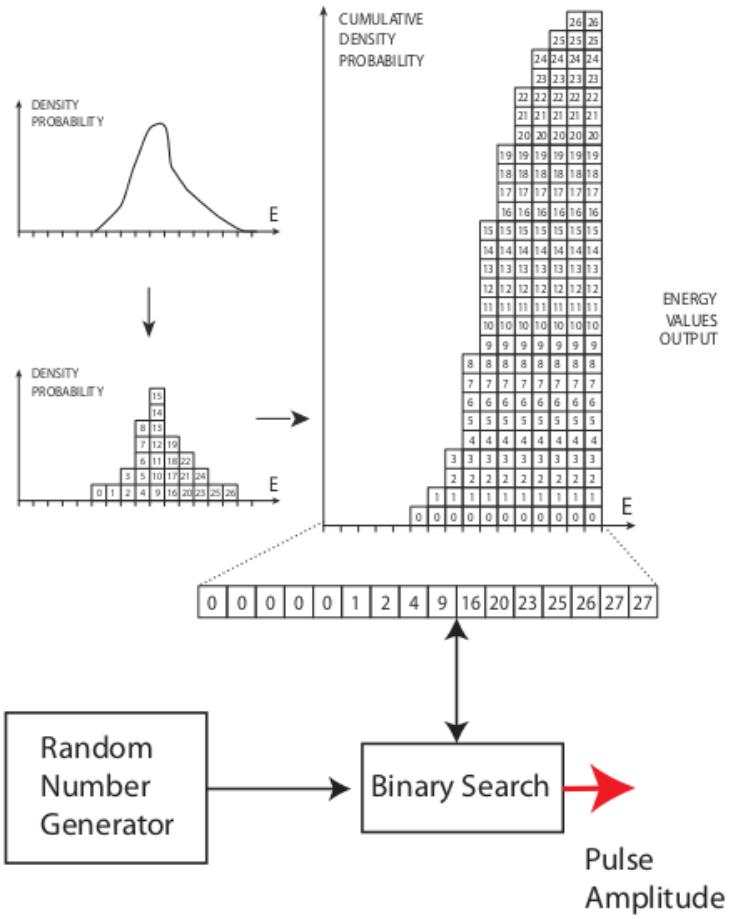

Figura 3.3: *Procedimento di emulazione della statistica energetica.*

Con riferimento alla figura 3.3 ogni quadrato di ciascuna colonna è numerato in sequenza. Consideriamo il caso semplificato in cui il numero totale di quadrati sotto la curva è una potenza di due, per esempio $2^5 = 32$. Supponiamo di usare un generatore di numeri casuali equiprobabili fino al valore massimo 32; i numeri casuali mappano completamente l'area sottesa dalla curva dello spettro. Ogni volta che viene generato un numero lo si ricerca nello spettro ed una volta trovato si estrae il numero del campione n indicando così il corrispondente valore di energia E_n. Se consideriamo nuovamente un generico campione x alto il doppio rispetto ad un campione y si vede

facilmente che, poiché i numeri casuali mappano tutti i quadrati con uguale probabilità, vi è una probabilità doppia che il numero casuale appartenga alla colonna di x piuttosto che a quella di y.

Nella pratica si impiega uno spettro di energia cumulativa $H_c(E)$ che viene calcolato a partire dallo spettro di energia $H(E)$ nel seguente modo: $H_c(E_i) = \int_0^{E_i} H(E)dE$.

Usando lo spettro cumulativo è ancora possibile identificare il campione che contiene il numero casuale generato attraverso un'estensione dell'algoritmo descritto. Per esempio (figura 3.3) , se il numero casuale è 18 si vede che esso appartiene alla cella di memoria numero 10. Infatti la cella 10 contiene un numero che è maggiore di 18 (20) mentre la cella precedente contiene un numero inferiore (16); ciò indica che il campione numero 10 contiene i quadrati che vanno da 16 a 19 esattamente il range cui 18 appartiene. Per questo motivo il valore di energia di uscita corrispondente al numero casuale 18 è 10.

3.2 Emulazione posizione di incidenza

Come per l'emulazione dello spettro di energia, l'emulazione della posizione di incidenza modifica l'ampiezza degli impulsi. In particolare per ogni canale viene calcolato un valore di modulazione che permette di scalare opportunamente le ampiezze degli impulsi. Lo schema riportato in figura 3.4 mostra il funzionamento di questo meccanismo.

Per spiegare come questo possa essere realizzato supponiamo di avere a disposizione una vera immagine tomografica. Attraverso software di imaging è possibile convertire l'immagine in una matrice i cui valori rappresentano la tonalità di colore. Sappiamo che l'immagine CT attraverso il colore esprime la quantità di radiazione rivelata in ogni punto del piano di osservazione della gamma camera. Prendiamo in considerazione la figura 3.5.

Come si vede un'immagine di NxN pixel raffigurante due spot luminosi è stata rappresentata in 3d. Un'altezza della curva maggiore indica, chiaramente, una maggior quantità di colore. Se pensiamo che questa immagine sia ottenuta da un esperimento tomografico abbiamo che gli spot luminosi

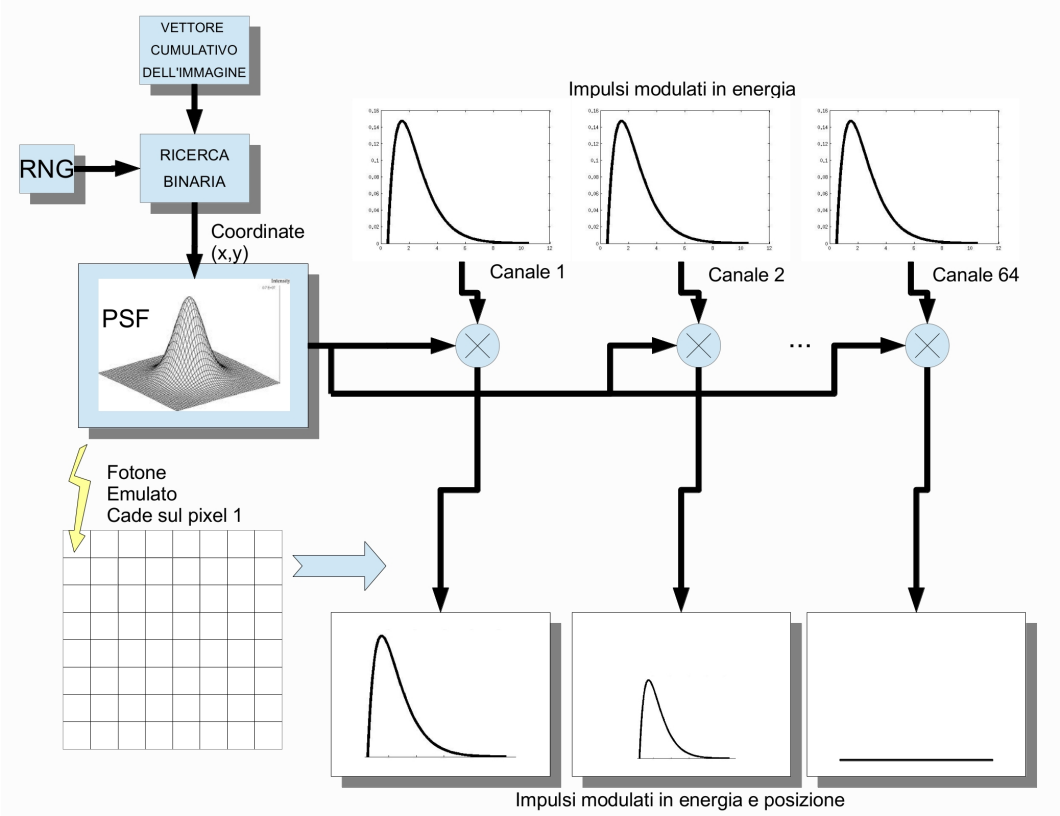

Figura 3.4: *Emulazione della posizione.*

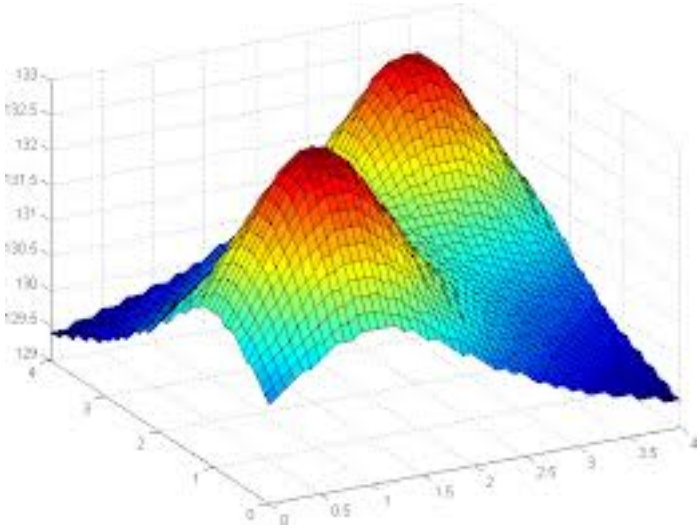

Figura 3.5: *Immagine rappresentante due spot luminosi.*

sono dovuti ad una esposizione maggiore alla radiazione rispetto al resto dell'immagine stessa. Possiamo cioè affermare che, preso un certo intervallo di tempo di osservazione corrispondente alla durata dell'esperimento, un gran numero di radiazioni hanno inciso in corrispondenza degli spot; l'immagine è cioè una rappresentazione a regime di una moltitudine di interazioni. Supponiamo ora di voler ricreare la stessa immagine pixel dopo pixel. Un fotone che incide in una certa posizione $\overline{x,y}$ ha come effetto nell'immagine finale un aumento nella tonalità di colore di una certa quantità $d\alpha$, dove $d\alpha$ è semplicemente il rapporto tra il colore più scuro ottenibile, ad esempio 1, ed il numero totale T di fotoni gamma osservati. Partendo da un'immagine completamente bianca è necessario scegliere la posizione del primo fotone, del secondo e dei successivi T-2 fotoni. Anche in questo caso risulta conveniente utilizzare una variante dell'algoritmo illustrato in precedenza per l'emulazione dello spettro di energia[1]. Infatti, immaginando di scomporre l'immagine esattamente in N righe di N pixel partendo dalla prima riga, ciascuna di queste può essere vista al pari di uno spettro di energia discretizzato. I valori dei pixel indicano la probabilità che un fotone incida nell'area competente al pixel stesso. Utilizzando un generatore di numeri casuali ricostruiamo l'immagine in questo modo.

Per fissare le idee ipotizziamo che l'immagine sia composta da 256x256 pixel. Numeriamo in modo crescente ciascun pixel partendo dal primo in alto a sinistra e procedendo dapprima lungo la riga superiore; Il pixel in basso a destra sarà il numero 65536. A questo punto disponiamo in un vettore di lunghezza 65536 tutti i pixel numerati in ordine; questo vettore è l'immagine srotolata. In un nuovo vettore della stessa dimensione del primo, detto cumulativo, inseriamo dei valori così ottenuti: La posizione N-esima contiene il valore della posizione N-1 esima sommato al valore del pixel N appartenente al vettore immagine srotolata. Così, ad esempio, la prima posizione contiene il valore del primo pixel dell'immagine srotolata, il secondo la somma del primo e del secondo pixel dell'immagine srotolata, il terzo la somma del primo del secondo e del terzo e così via. Facciamo notare che prese due posizioni consecutive nel vettore cumulativo, la differenza dei

[1]Ideato dall'ing. Andrea Abba.

valori contenuti in esse, rappresenta la probabilità della posizione ad indice maggiore ovvero la probabilità del pixel di essere investito dalla radiazione come verrà ora illustrato.

Alla fine del processo di creazione del vettore cumulativo l'ultimo elemento potrà contenere un valore che è al massimo $256*256*T*d\alpha = MAX$ avendo supposto che tutti i pixel della matrice siano stati esposti omogeneamente a radiazione. In un caso più generale invece MAX dipenderà dall'immagine. Imponiamo che questo valore sia il fondo scala per il generatore di numeri casuali ottenendo il minimo valore estraibile pari a 0 ed il massimo pari a MAX. Facciamo notare che ciò è facilmente ottenibile anche in presenza di un generatore di numeri casuali con estremi fissi in quanto tutti i valori del vettore cumulativo possono essere scalati opportunamente in modo tale che MAX coincida con l'estremo massimo di estrazione. Come si intuisce questa operazione è di fondamentale importanza in quanto si vuole rimappare con omogeneità lo spazio dei valori uniformemente distribuiti generati dal *RNG* con il dominio della funzione F che realizza la conversione spettrale vista in precedenza. La funzione F, la cui uscita non è nient'altro che l'immagine sintetizzata, si può implementare in questo modo.

Fornito un numero casuale se ne effettua la ricerca all'interno del vettore cumulativo. Ovviamente vista la costruzione del vettore è molto probabile che il numero non sia presente. Interessa però individuare la posizione nel vettore cui corrisponde il valore più prossimo a quello ricercato e superiore ad esso.

Scansioniamo il vettore cumulativo partendo dalla prima posizione; se il valore contenuto in essa è inferiore al numero estratto si passa al secondo elemento e la ricerca continua. La ricerca si interrompe quando si trova un valore appena superiore a quello cercato. Ne consegue che la posizione precedente a quella trovata contiene il valore inferiore più vicino a quello estratto dal *RNG*. L'insieme delle posizioni trovate in seguito a ripetute estrazioni e ricerche rispetta la statistica dell'immagine; Ognuna di queste indica, per via della corrispondenza con il pixel associato, la posizione estratta di incidenza della radiazione. E' importante evidenziare che è più probabile estrarre un numero associato ad un pixel ad elevato colore. Il vettore cumulativo

è visualizzabile come una gradinata (fig 3.6) con differenze di altezze tra i gradini tanto maggiori quanto più il pixel associato è probabile. Un gradino associato al pixel X con ampiezza maggiore fa si che il numero estratto molto probabilmente appartenga ad esso rispetto ad un altro con altezza minore.

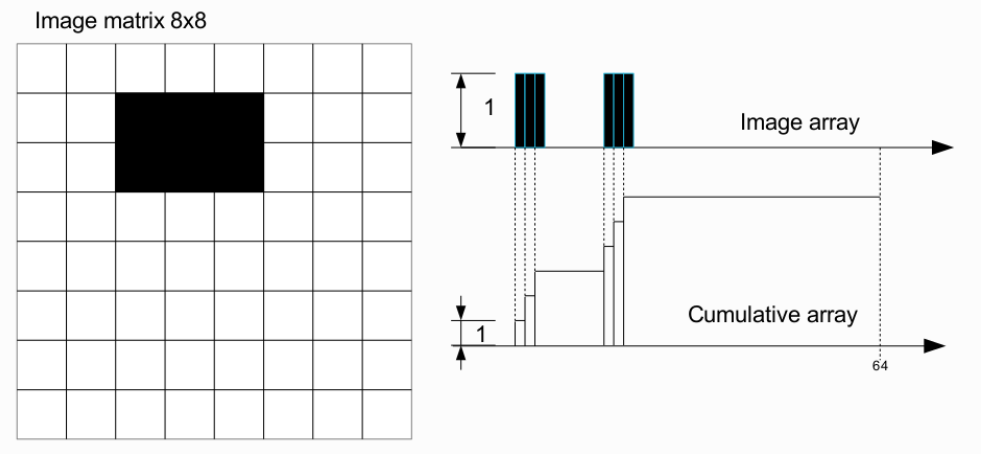

Figura 3.6: *Conversione di un'immagine di 64 pixel nel vettore cumulativo.*

3.2.1 PSF

Abbiamo sottolineato che ogni interazione gamma sullo scintillatore produce, attraverso i meccanismi illustrati nel capitolo 2, una luminescenza diffusa. Ciò vuol dire che la carica raccolta dai rivelatori in seguito ad un evento tiene conto della produzione di un gran numero di fotoni ottici. Dal punto di vista dell'emulazione possiamo ipotizzare che ad ogni fotone incidente sia associata una certa carica totale Q in uscita. Essa sarà pertanto costituita dalla sommatoria delle cariche parziali competenti a ciascun detector eccitato in seguito all'evento di interazione.

Nella realtà si osserva che la distribuzione di intensità luminosa z prodotta, ad esempio, da un'interazione al centro del cristallo, assume l'andamento mostrato in figura 3.7.

Si riconosce una forma gaussiana bidimensionale di espressione:

$$z(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}((\frac{x}{\sigma_x})^2+(\frac{y}{\sigma_y})^2)}$$

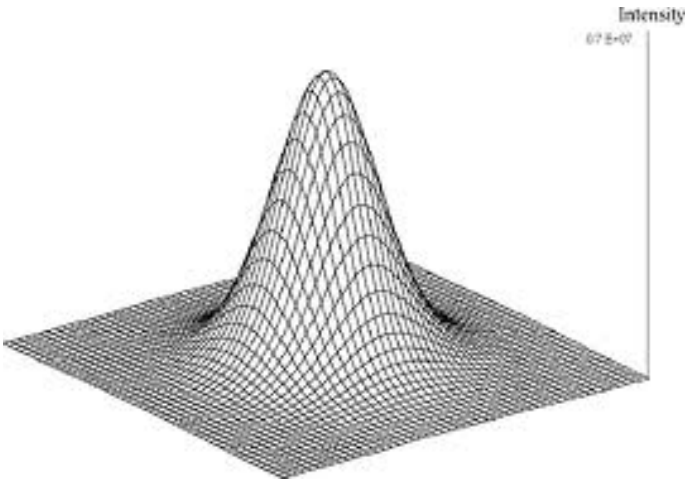

Figura 3.7: *Distribuzione gaussiana.*

Per tener conto di questo fenomeno di diffusione luminosa ricorriamo al concetto di PSF. Prendiamo in considerazione una matrice di N*N rivelatori. Ad ognuno di questi rivelatori associamo una Point Spread Function (PSF) che ci fornisce l'informazione relativa alla carica competente a ciascun rivelatore. Le PSF sono personalizzate per ogni rivelatore e vengono calcolate a partire dalla distribuzione gaussiana bidimensionale allineando quest'ultima in corrispondenza del centro di ogni detector. Tipicamente la risoluzione della PSF, ovvero il numero di punti di cui è composta, coincide con quella della matrice immagine di partenza. In questa condizione il valore della PSF i-esima in corrispondenza delle coordinate di un generico pixel estratto dall'algoritmo di cui sopra fornisce immediatamente il valore di carica prodotta dal detector i-esimo. Pertanto per ogni coordinata estratta tramite N*N ricerche si ottengono i valori di carica di tutti i detector. Questi valori, raccolti nella matrice "charge", servono per modulare l'ampiezza degli impulsi non appena viene triggerato un evento.

3.2.2 Test con Matlab

Come primo studio di fattibilità del sistema si è fatto uso del programma di calcolo scientifico Matlab.

Il codice può essere scomposto in tre parti. Nella prima viene sintetizzata

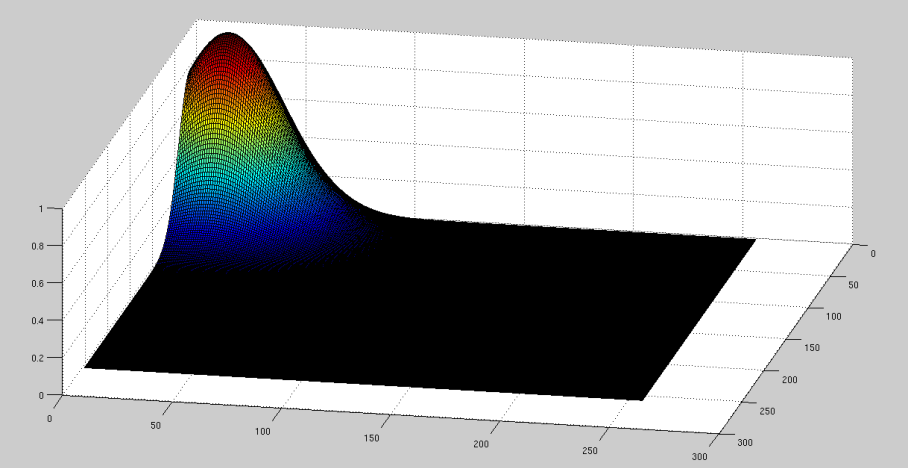

Figura 3.8: *PSF per un detector angolare.*

un'immagine arbitraria (vedi figura) e si costruisce il vettore cumulativo necessario nel seguito dell'elaborazione. Nella seconda si ha l'emulazione vera e propria ovvero l'estrazione dei numeri casuali ed il conseguente calcolo delle cariche dei detector. Nella terza si effettua una ricostruzione dell'immagine originaria per verificare la correttezza del procedimento.

L'estrazione di numeri casuali avviene attraverso la funzione di libreria rand(). L'implementazione di tale funzione non risulta particolarmente vincolante purché siano rispettate le condizioni di pseudo casualità dei numeri estratti. Una possibile implementazione ricorre al metodo matematico congruenziale polinomiale.

In matematica un generatore lineare congruenziale (LCG, dall'inglese Linear Congruential Generator) è un algoritmo per la generazione di numeri pseudo-casuali vecchio e molto conosciuto. La teoria sulla quale poggia è semplice da capire e da implementare; inoltre ha il vantaggio di essere computazionalmente leggero. Il generatore è definito ricorsivamente: $X_{n+1} = (aX_n + c) : m$ dove X_n è un valore della successione dei numeri pseudocasuali, m>0 è il modulo, a è una costante moltiplicativa compresa tra zero ed m, c è l'incremento compreso tra zero ed m (zero nel caso speciale in cui si parla di generatore Park-Miller RNG o moltiplicativo) ed X_0viene detto seme della successione o valore iniziale. L'operatore : restituisce il resto intero della divisione tra $aX_n + c$ ed m. Il periodo di un LCG è al più m, e

per alcune scelte di a può essere molto più piccolo. Nonostante gli LCG siano generalmente in grado di produrre numeri pseudocasuali di buona qualità, la loro qualità è molto sensibile alla scelta dei coefficienti c, m ed a.

Storicamente, scelte sbagliate hanno portato a implementazioni inefficienti di LCG. Un esempio significativo è RANDU che è stato usato largamente nei primi anni 1970 e ha portato a dei risultati che attualmente sono messi in dubbio a causa dell'uso di un LCG di scarsa qualità.

Se un generatore congruenziale lineare è inizializzato con un carattere e iterato, il risultato è un semplice cifrario classico chiamato cifrario affine; questo cifrario è facilmente forzabile con un'analisi frequentista standard. I generatori congruenziali lineari richiedono una memoria minima (tipicamente 32 o 64 bit) per memorizzare lo stato. Gli LCG non dovrebbero essere usati in applicazioni dove è critico l'utilizzo dell'alta casualità. Per esempio, non sono adatti per simulazioni Monte Carlo poiché hanno una correlazione tra di loro e non devono essere usati in crittografia. Gli LGC possono essere una buona scelta in alcuni contesti, come per esempio nei sistemi embedded, dove la memoria disponibile è spesso molto limitata.

Per ciò che concerne la ricerca all'interno del vettore cumulativo risulta conveniente l'approccio della ricerca binaria.

Immaginiamo di avere dei dati ordinati, numeri, oppure stringhe. Per esempio supponiamo di dovere cercare una parola in un dizionario. Un modo potrebbe essere quello di cominciare dalla lettera a e andare avanti fino a che non troviamo la parola. Questa ricerca si chiama ricerca sequenziale o ricerca lineare. Se la parola è "zuzzerellone", la nostra ricerca non è stata molto intelligente. Un metodo più comunemente usato è il seguente. Apriamo il dizionario in qualunque punto e quindi andiamo avanti o indietro a seconda della parola cercata, visto che il dizionario è ordinato alfabeticamente.

La ricerca binaria è una implementazione di questo metodo. Per semplificare, supponiamo di volere cercare un numero in un vettore ordinato di numeri. Troviamo l'elemento medio del vettore, se tale elemento è l'elemento cercato la ricerca è conclusa, altrimenti dobbiamo verificare se il numero cercato è maggiore o minore dell'elemento medio.

Se è maggiore cercheremo nella metà superiore, se è minore cercheremo

nella metà inferiore del vettore. Questo processo va ripetuto finché il numero non viene trovato. Nella ricerca binaria, ad ogni passo dimezziamo la dimensione del vettore in cui cercare il numero, pertanto se il vettore è di dimensione n, nel peggiore dei casi dovremo analizzare solo $log_2 n$ elementi. Osserviamo che $log_2 n$ indica il numero di volte che n deve essere diviso per due per arrivare ad 1. Quindi se abbiamo, come nel caso del vettore cumulativo, una collezione di n = 65536 oggetti, una ricerca sequenziale, nel peggiore dei casi deve visitarli tutti e, quindi, farà n confronti mentre la ricerca binaria ne fa al massimo 16.

3.2.3 Ricostruzione immagine

La sezione relativa alla ricostruzione dell'immagine ci permette di stimare il numero di simulazioni necessarie per avere una buona rappresentazione dell'immagine di partenza. Raccogliendo 650000 matrici charge si ottiene decisamente una buona ricostruzione dell'immagine come possiamo vedere in figura 3.9.

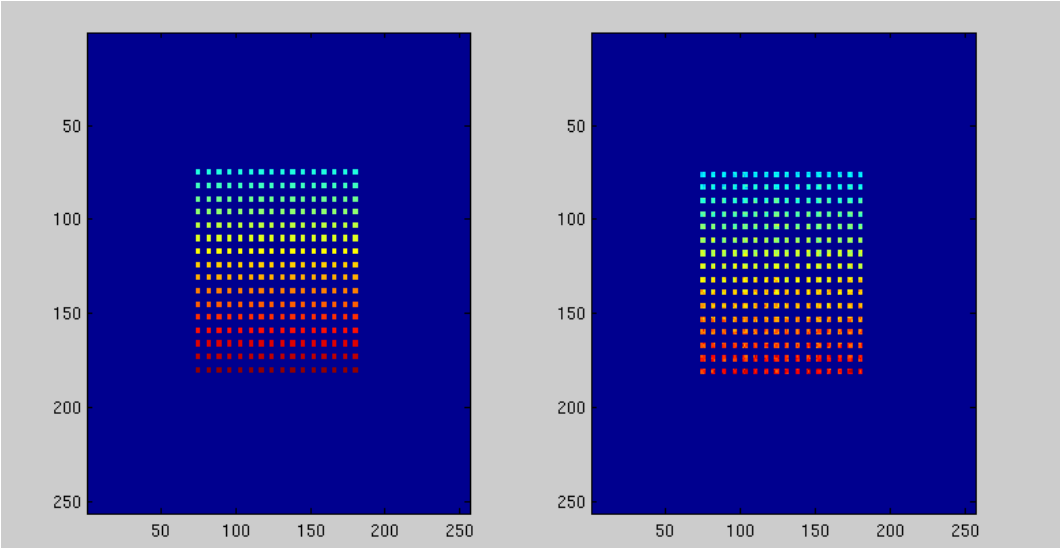

Figura 3.9: *A sinistra l'immagine di test, a destra quella simulata.*

A sinistra viene mostrata l'immagine arbitraria di partenza, che rappresenta dei quadrati di intensità luminosa omogenea a righe su sfondo scuro.

Affinché fosse semplice la percezione visiva del risultato di ricostruzione si è deciso di rappresentare la figura a colori, sebbene in realtà l'algoritmo lavori su un'immagine in bianco e nero, per cui il colore è da intendersi indice di un certo livello nella scala dei grigi. I quadrati rappresentano le uniche zone colpite da radiazione. Al sinistra si ha l'immagine corrispondente alla reale posizione di incidenza avuta durante l'emulazione. Nell'immagine 3.10, invece, si illustra l'immagine ricostruita, a partire dalle 650000 matrici charge, attraverso il metodo del baricentro.

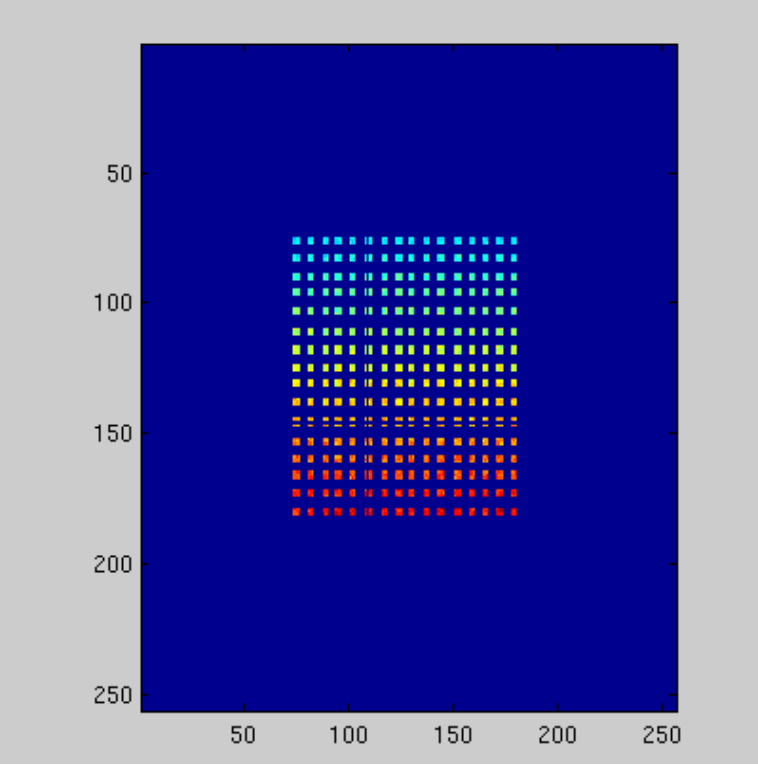

Figura 3.10: *Immagine ricostruita a partire da quella di test.*

Il metodo del baricentro è un metodo di media che permette di trovare la posizione dell'interazione attraverso interpolazione.

E' noto che una media pesata (o ponderata) di n elementi ha espressione:

$$M = \sum_{i=0}^{n-1} \alpha_i x_i$$

dove α_i è il peso i-esimo del campione x i-esimo. Se i pesi sono costanti si parla di media aritmetica semplice tra numeri ottenuta sommando ogni numero e dividendo il risultato per il numero totale di elementi su cui si sta mediando. Nel nostro caso i coefficienti peso sono rappresentati dai valori contenuti nella matrice charge per ogni estrazione. In effetti il problema della media bidimensionale può essere sdoppiato in due sottoproblemi monodimensionali. Prendiamo, dunque, in esame il calcolo della sola posizione x in un sistema di assi cartesiani ortogonali x, y. Considerando di suddividere in tre parti uguali un segmento di lunghezza 30 immaginiamo che ognuno di essi rappresenti l'estensione di un detector, sulla matrice, in un'unica dimensione; se il detector è "lungo" 10 esso nella realtà occuperà un'area di 10 x 10. Come mostra la figura 3.11 ci è utile visualizzare il peso per ciascun detector su un asse$z$. Nel caso specifico il primo detector ha peso 3 il secondo 10 e

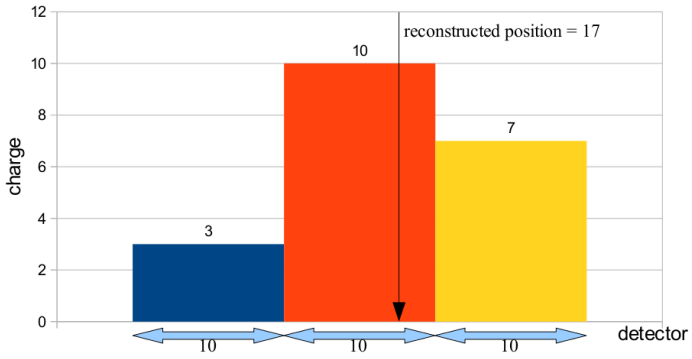

Figura 3.11: *Rappresentazione monodimensionale di una matrice charge.*

il terzo 7. Intuitivamente si può presumere che la radiazione gamma abbia inciso in una zona molto prossima al detector centrale e verso la sua destra per via della maggior carica raccolta dal detector di destra rispetto a quello di sinistra. Numericamente si calcola la coordinata:

$$\overline{x} = \frac{3}{20}5 + \frac{10}{20}15 + \frac{7}{20}25 = 17$$

Notiamo che il peso di ciascun detector è associato con la coordinata corrispondente al centro dello stesso. Questa assunzione è giustificata dal fatto che a priori non si può sapere se tutti i punti del detector abbiano contribui-

to allo stesso modo alla carica totale. Dunque, effettuando la media su un segmento di lunghezza 10 si ottiene semplicemente il valor medio 5.

Naturalmente questo tipo di media non è l'unico possibile sebbene fornisca una buona ricostruzione, come dimostrato della simulazione.

Il procedimento appena descritto si estende ad un numero arbitrario di detector, nello specifico 8, e si ripete anche per la coordinata y. Come illustrato in precedenza ogni posizione calcolata contribuisce alla costruzione dell'immagine finale con un contributo $d\alpha$.

A questo punto appare chiaro come sia possibile avendo a disposizione una gamma camera con un numero limitato di detector riuscire ad avere immagini con buona risoluzione. Nel progetto si è deciso di utilizzare la dimensione canonica di 256x256 pixel.

Un'inconveniente del processo di media è l'effetto di bordo. Quando un fotone incide molto vicino al perimetro della matrice molti dei fotoni ottici generati vengono persi in quanto non sono presenti abbastanza detector. Utilizzando l'algoritmo di media appena descritto qualsiasi fotone incidente molto vicino al perimetro viene ricostruito in una posizione più interna di quella effettiva. Questo è dovuto al fatto che i detector hanno peso omogeneo in tutta la matrice e non esistono abbastanza "campioni di bordo" su cui mediare.

3.3 Emulazione statistica temporale

L'emulazione di una sorgente di radiazione richiede un'accurata simulazione dei tempi di emissione. Sebbene sia possibile emulare gli istanti di emissione considerando soltanto il tasso medio degli eventi, sarebbe opportuno, ai fini di una più accurata emulazione, convertire una distribuzione arbitraria di eventi (quale quella Poissoniana vista nel precedente capitolo), in un vettore di impulsi di start. L'algoritmo che genera l'istante di emissione degli impulsi, sfrutta la distribuzione dell'intervallo temporale di emissione di due fotoni consecutivi per generare un impulso di sincronismo che opera similmente ad un segnale di interrupt all'interno di un sistema di calcolo.

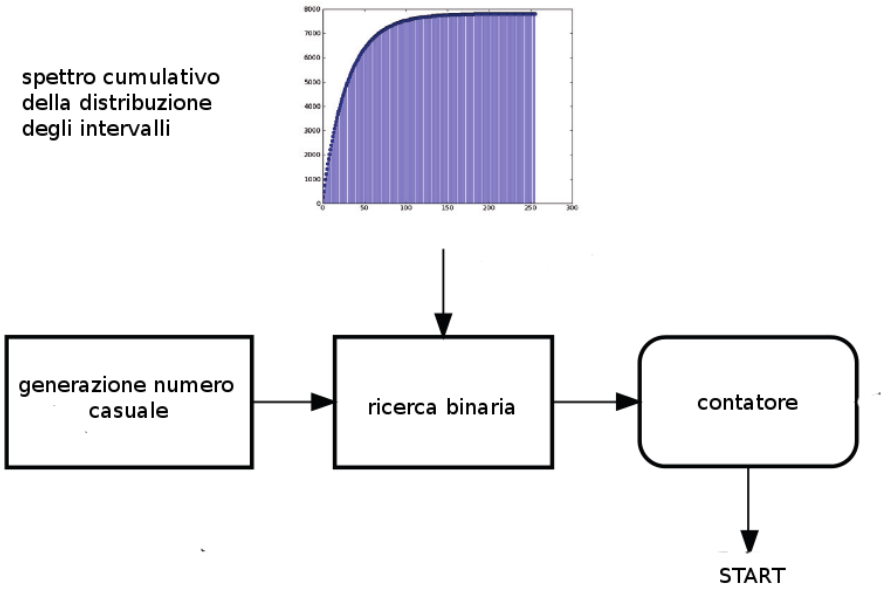

Figura 3.12: *Funzionamento dell'emulazione della statistica temporale.*

La distribuzione arbitraria utilizzata è quella derivata dall'andamento Poissoniano ovvero

$$I(t)dt = re^{-rt}dt$$

che rappresentiamo in figura 3.13.

L'algoritmo produce un vettore di intervalli temporali $\triangle\ t$ che vengono convertiti in un segnale di start ad 1bit. La generazione dei tempi di occorrenza viene realizzata analogamente a quanto visto per i valori di ampiezza nell'emulazione dello spettro di energia utilizzando semplicemente una differente statistica in ingresso.

Di fatti, i tempi di arrivo sono utilizzati per calcolar il tempo intermedio tra impulsi di start consecutivi che, per una sorgente ad emissione Poissoniana, hanno una distribuzione di probabilità esponenziale con costante di tempo rappresentante il tempo medio fra due impulsi e il suo inverso il tasso medio di impulsi.

Quindi un blocco del tutto identico al generatore delle ampiezze ma alimentato con uno spettro esponenziale può essere utilizzato per generare una sequenza di impulsi coerenti con la distribuzione Poissoniana ed al rate desi-

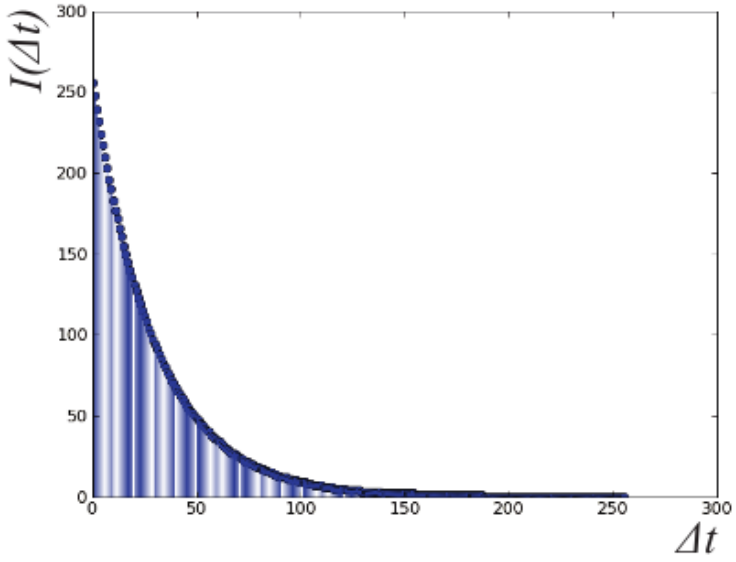

Figura 3.13: *Probabilità di avere due eventi consecutivi in un intervallo Δt.*

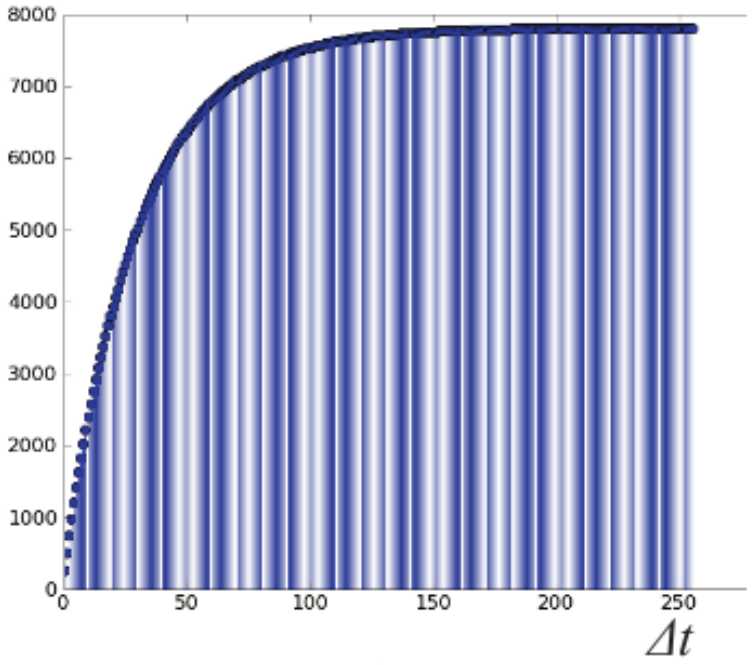

Figura 3.14: *Versione cumulativa di $I(\Delta t)$.*

derato. Se si volesse utilizzare una statistica differente basterebbe cambiare la distribuzione fornita al sistema. Per esempio volendo generare un rate costante di emissione basterebbe utilizzare una distribuzione a un solo campione coincidente appunto con il rate desiderato

Gli impulsi di start sono generati mediante un semplice contatore. Quando il sistema si avvia viene calcolato un intervallo temporale cioè un valore che può essere programmato nel contatore e, successivamente, si genera un impulso di start. Non appena viene emulata l'emissione del primo fotone attraverso l'impulso di start il contatore viene programmato con il valore appena calcolato ed ha inizio un conteggio decrescente. Simultaneamente un nuovo valore di intervallo è pronto per riprogrammare il contatore cioè subito dopo che il conteggio del primo valore ha raggiunto lo zero. In quell'istante si ha la generazione di un nuovo segnale di start ed il meccanismo si ripete.

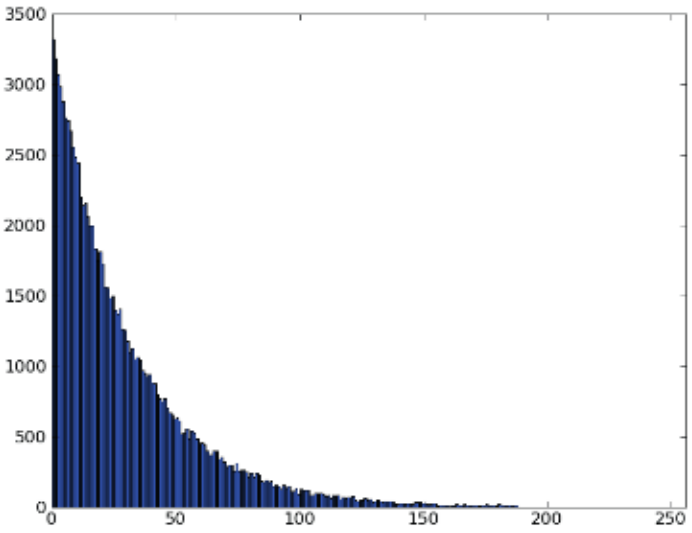

Figura 3.15: *Distribuzione dei Δt emulati.*

Anche in questo caso è stato utilizzato uno script Matlab per validare la procedura. Una versione cumulativa della distribuzione degli intervalli temporali viene calcolata a partire dalla distribuzione stessa. Il risultato della simulazione è un vettore di 100000 intervalli temporali, che rispettano come è possibile vedere in figura 3.15 la statistica di partenza.

3.3.1 Prove di Bernoulli

Il procedimento appena illustrato per la generazione degli impulsi di START può essere notevolmente semplificato. Nel caso particolare in cui la statistica di emissione temporale sia di tipo Poissoniano è possibile generare con estrema facilità la sequenza di impulsi. Consideriamo di generare ad un rate costante numeri casuali uniformemente distribuiti in un intervallo $0 - \gamma$. Fissata una certa soglia β all'interno di questo intervallo, decidiamo di lanciare un nuovo impulso di START ogni volta che il valore estratto è inferiore alla soglia; in questo modo si ottiene una sequenza di impulsi aderente alla statistica Poissoniana.

Per spiegare come questo sia possibile facciamo notare che il meccanismo appena illustrato corrisponde ad effettuare un grande numero di prove di Bernoulli ove ciascuna di esse è un esperimento statistico in cui il risultato può essere o *successo* o *fallimento.* Nel nostro caso, infatti, si ha un successo, cioè un impulso di START, quando il valore casuale è inferiore ad una certa soglia ed un fallimento nel caso contrario.

Dimostriamo quindi come derivare una distribuzione Poissoniana sfruttando successive prove di Bernoulli.

Ricordiamo che il numero di eventi accaduti in un intervallo qualsiasi, ad esempio l'intervallo $0 - T$ che indichiamo con $X(T)$, è distribuito secondo Poisson se sono verificate le seguenti ipotesi:

1. Il numero di eventi in un certo intervallo è indipendente dal numero di eventi in un intervallo di tempo qualsiasi, distinto dal precedente.

2. Per ogni intervallo $\triangle T$, per quanto piccolo, c'è una probabilità $\lambda \triangle T$, dove λ è una costante positiva, che si presenti un evento.

3. Per intervalli abbastanza piccoli può verificarsi al massimo un solo evento.

Per derivare l'espressione della densità di Poisson, in base alle ipotesi precedenti, si suddivide l'intervallo T in $n = \frac{T}{\triangle T}$ intervalli di ampiezza $\triangle T$. La presenza o l'assenza di un evento in ogni $\triangle T$ è un evento di Bernoulli in n prove a cui è associata un certa probabilità $p = \lambda \Delta T$. Facendo tendere ΔT

a zero ed n all'infinito per avere un intervallo infinitesimo, in accordo con la 2, otteniamo[2]:

$$P\{\mathrm{X(T)=k}\} = \lim_{\Delta T\to 0, n\to\infty} \binom{n}{k}(p)^k(1-p)^{n-k} =$$

$$= \lim_{\Delta T\to 0, n\to\infty} \binom{n}{k}(\lambda\Delta T)^k(1-\lambda\Delta T)^{n-k}$$

dove $P\{\mathrm{X(T)=k}\}$ è la probabilità di avere k successi in n prove. Sviluppando otteniamo:

$$= \lim_{\Delta T\to 0, n\to\infty} \binom{n}{k}(\frac{\lambda T}{n})^k(1-\frac{\lambda T}{n})^{n-k} =$$

$$= \frac{(\lambda T)^k}{n!} \lim_{\Delta T\to 0, n\to\infty} [n(n-1)...\frac{(n-k+1)}{n^k}](1-\frac{\lambda T}{n})^{n-k}$$

I termini fra parentesi quadre sono k ed il rapporto $\frac{(n-k+1)}{n^k}$ tende ad 1 per n tendente all'infinito. Inoltre ricordando il limite notevole

$$\lim_{n\to\infty}(1-\frac{\lambda T}{n})^{n-k} = e^{-\lambda T}$$

si ottiene:

$$P(k) = \frac{(\lambda T)^k}{k!}e^{-\lambda T}$$

ovvero la distribuzione di Poisson.

3.4 Emulazione forma dell'impulso

Il blocco di emulazione della forma d'onda dell'impulso permette di impiegare diverse forme d'onda pre-programmate. Una forma d'onda viene trasmessa lungo la catena di emulazione non appena si riceve un segnale di start dall'emulatore della statistica temporale. L'uscita del generatore di forma d'onda

[2]Dalle prove di Bernoulli la probabilità che in n tentativi si abbiano k successi ed n-k fallimenti è data dall'espressione $\binom{n}{k}(p)^k(1-p)^{n-k}$ dove il coefficiente binomiale tiene conto di tutte le possibili permutazioni dei successi e fallimenti.

viene "modulata" dall'emulatore dello spettro energetico per riprodurre la statistica energetica desiderata. Successivamente si ha un'ulteriore modulazione per associare alla forma d'onda l'informazione relativa alla posizione di scintillazione emulata.

La forma d'onda può essere descritta analiticamente o registrata attraverso un setup sperimentale impiegando una opportuna catena di acquisizione.

I compiti del generatore di forma d'onda sono:

- Leggere la memoria che contiene la forma dell'impulso
- Riconoscere quando l'ampiezza del segnale è inferiore ad una certa soglia così da poter ritenere terminato il segnale
- Interpolare l'uscita della memoria.

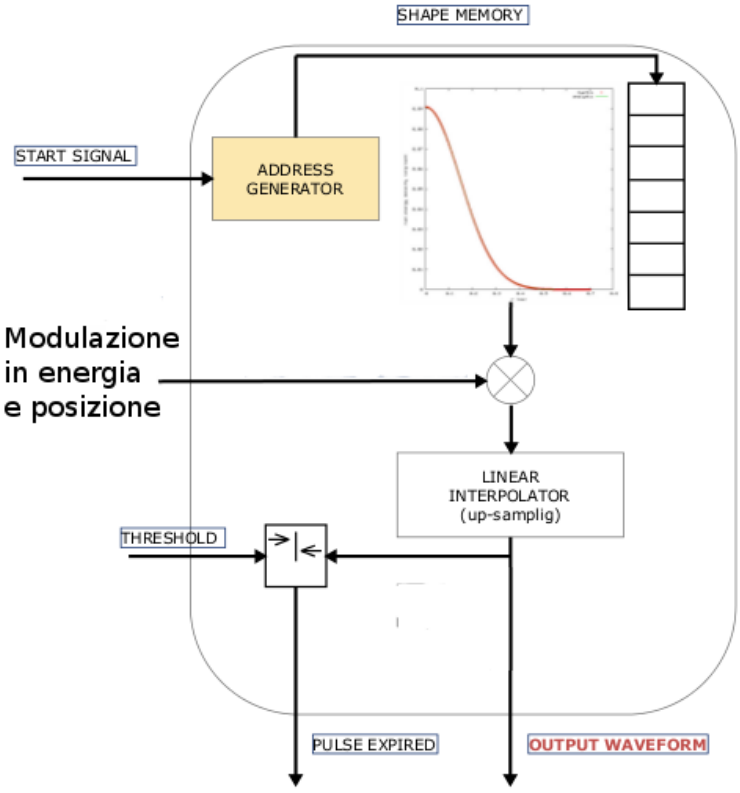

Figura 3.16: Struttura dell'emulatore di forma d'onda

La figura 3.16 mostra l'architettura del generatore di forma d'onda. Una vettore quantizzato in tempo che rappresenta la forma dell'impulso è salvato in una memoria e viene letto dal primo all'ultimo elemento ogni volta che si ha un segnale di start. Leggendo le celle di memoria ad un rate costante che alimenta lo stadio di conversione digitale analogico si ha la ricostruzione dell'onda; l'emulatore simula il segnale di uscita della gamma camera.

Il rate di lettura della memoria dipende dall'abilitazione dell'interpolazione. L'interpolazione consente la riduzione della memoria necessaria per salvare una certa forma d'onda. Per spiegare come funziona l'interpolatore consideriamo di dover ricostruire il segnale esponenziale visto al capitolo 2 con una lunga costante di tempo di 10ms. Leggendo ad esempio la memoria ad un rate di 250Msample/s sarebbero necessarie 2,5M parole di memoria. Utilizzando un interpolatore 1:1000 si avrebbe una riduzione dello spazio occupato notevole poiché sarebbero necessarie solo 2,5K parole. Al fine di minimizzare la complessità dell'interpolatore è stato deciso di utilizzare un interpolatore di tipo lineare. Leggendo due valori consecutivi nella memoria l'interpolatore in questione semplicemente stima la retta congiungente i due punti, rappresentati su un piano cartesiano, e calcola 1000 nuovi punti giacenti su di essa e reciprocamente equispaziati.

3.4.1 Pileup

Fin'ora è stata omessa nella descrizione del processo di emulazione una sofisticazione aggiuntiva che l'emulatore è in grado di gestire. Sappiamo che in corrispondenza di ciascun segnale di start la catena di emulazione risponde elaborando le forme d'onda dei 64 canali. Questo vuol dire che la lettura e l'elaborazione delle forme d'onda si estende per un certo tempo dopo il segnale di start. L'intervallo di tempo è quello necessario per riprodurre fedelmente l'intera forma d'onda, dal picco massimo alla coda esponenziale. Per spiegare il fenomeno del pileup immaginiamo che il generatore dei segnali di start lanci due eventi molto ravvicinati ovvero tali che il loro $\triangle\ t$ sia inferiore al tempo necessario per scansionare l'intera memoria della forma d'onda. Questo vuol dire che il sistema ha prodotto un nuovo fotone emulato

prima che il precedente abbia esaurito completamente il suo effetto. In uscita all'emulatore ci si aspetta di vedere questo fenomeno con la sovrapposizione del nuovo segnale al precedente; in altre parole il nuovo segnale si impila sulla coda esponenziale dell'impulso precedente. Per poter adeguatamente gestire questo fenomeno si ricorre alla struttura rappresentata in figura 3.17.

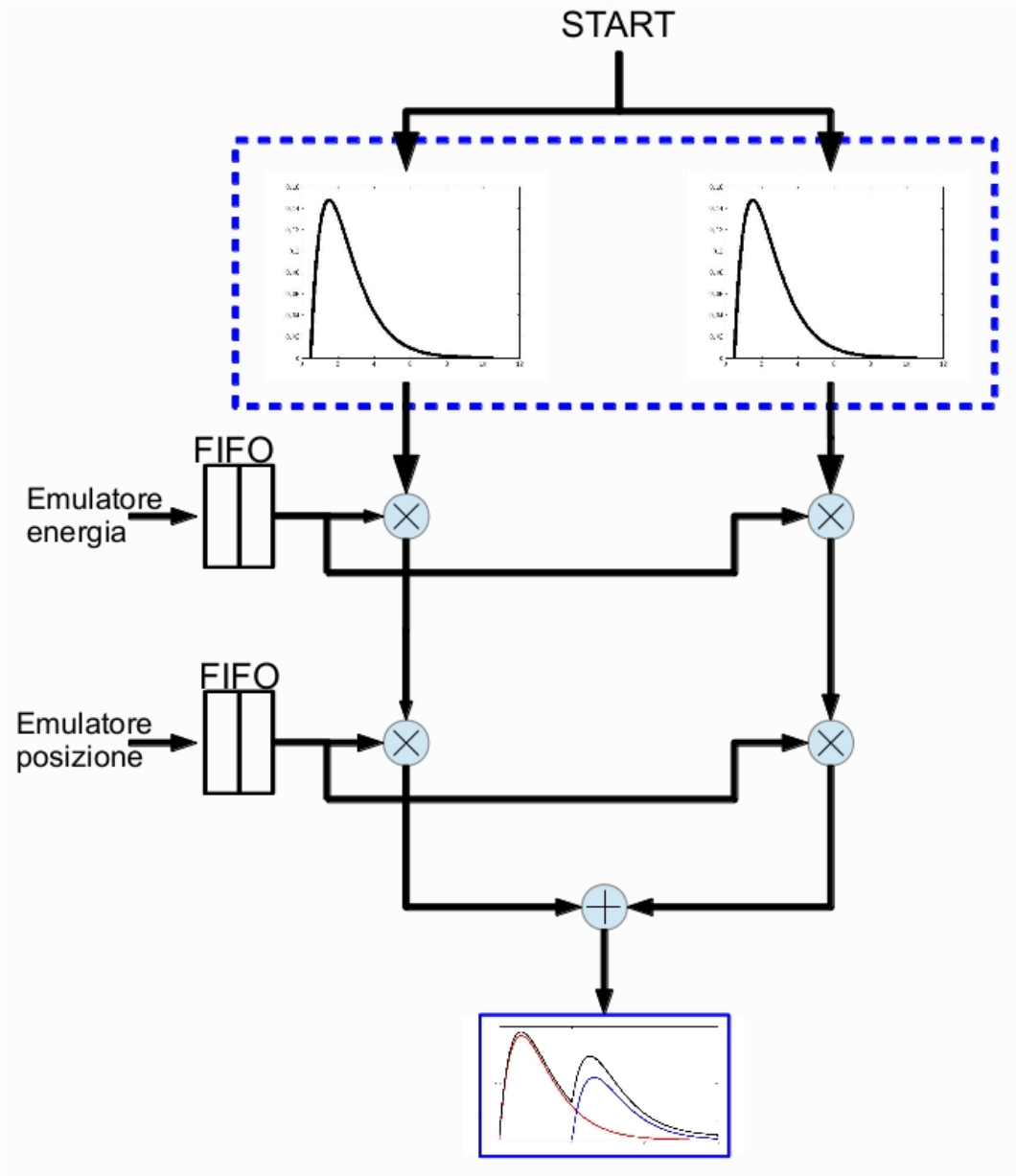

Figura 3.17: *Rappresentazione del fenomeno di pileup.*

Lo schema si riferisce ad un solo canale. Il primo segnale di start stimola la generazione della prima forma d'onda, ad esempio quella sulla sinistra, che viene modulata dal primo campione presente nelle due memorie FIFO. Il secondo segnale di start stimola invece la generazione di una seconda forma d'onda che diversamente dalla prima viene modulata dal secondo campione presente nelle due FIFO. Il nodo sommatore effettua il pileup cioè sovrappone i due segnali.

La struttura può essere espansa a piacere al fine di emulare un numero maggiore di eventi di pileup. Ad esempio con quattro catene di pileup basta utilizzare FIFO a 4 elementi ed un sommatore a 4 ingressi. Poiché la struttura presentata va replicata per tutti i 64 canali il numero di eventi di pileup che si possono emulare è limitato dal numero di percorsi paralleli che possono essere inseriti nell'implementazione del sistema cioè dalla quantità di risorse di cui si dispone.

3.4.2 Emulazione rumore

Questa funzione viene realizzata per ciascun canale attraverso un generatore di rumore che sintetizza uno stream di valori di rumore. Il rumore prodotto dal generatore va a sommarsi alla forma d'onda singola o in pileup (vedi fig 3.18) e produce il segnale finale che verrà effettivamente inviato allo stadio di conversione digitale/analogico.

Il rumore che si va a sintetizzare ha la caratteristica di avere una densità di probabilità gaussiana.

L'emulazione del rumore viene realizzata attraverso un semplice procedimento derivato come approssimazione pratica di uno dei più importanti teoremi di statistica e della teoria delle probabilità: il teorema del limite centrale.

Questo afferma che, considerata una sequenza di n variabili aleatorie X_1 X_2 X_3 ... X_n indipendenti, di valor medio m_k e varianza σ_k^2, la somma $Y = \sum_{k=1}^{n} X_k$ è una variabile aleatoria il cui valor medio e varianza sono dati da:

$$m = \sum_{k=1}^{n} m_k$$

$$\sigma^2 = \sum_{k=1}^{n} \sigma_k^2$$

La densità di probabilità è data dalla convoluzione delle densità di probabilità associate alle singole variabili aleatorie. Il teorema afferma che questa densità al tendere di n all'infinito e sotto condizioni molto generali (in particolare nessuna delle variabili aleatorie deve prevalere sulle altre nella somma) tende, almeno per un ampio intervallo intorno al valor medio (ossia nella parte centrale), ad una densità gaussiana con m e σ^2date dalle formule precedenti. L'approssimazione, per densità simmetriche, è già molto buona per n piccolo. Questo teorema riguarda una proprietà matematica della convoluzione di n funzioni positive e quindi non è legato esclusivamente alla statistica.

Nella pratica è possibile utilizzare variabili aleatorie con distribuzione uniforme. Per stimare un numero sufficiente di variabili da considerare nella sommatoria si è utilizzato un semplice script Matlab. Viene generato dapprima un vettore contenente un'insieme di valori uniformemente distribuiti e successivamente si genera un vettore di valori con distribuzione gaussiana attraverso una opportuna combinazione lineare:

$$y_i = x_{4i+1} - x_{4i+2} + x_{4i+3} - x_{4i+4}$$

Il risultato della simulazione con 500000 elementi estratti è visibile in figura 3.20: la curva rossa rappresenta la gaussiana che presenta ugual valor medio e varianza della gaussiana ottenuta col processo di combinazione lineare rappresentata invece in blu. Come si vede già con n=4 si ha un'ottima approssimazione.

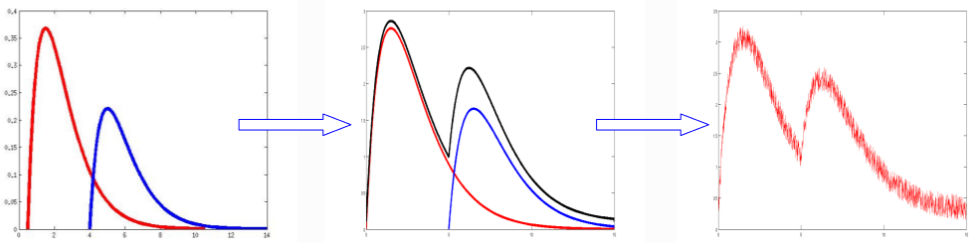

Figura 3.18: *Segnale finale per un canale che riceve due impulsi in pile up.*

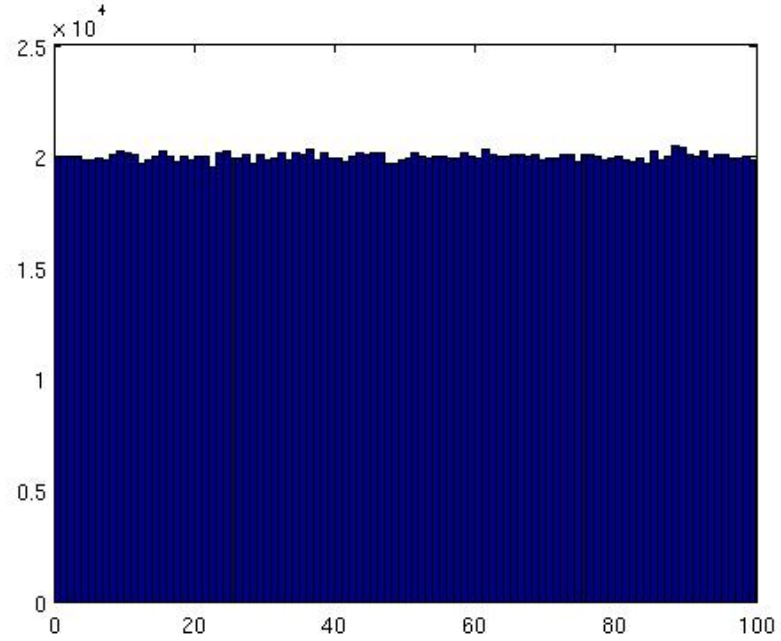

Figura 3.19: *Distribuzione uniforme di partenza.*

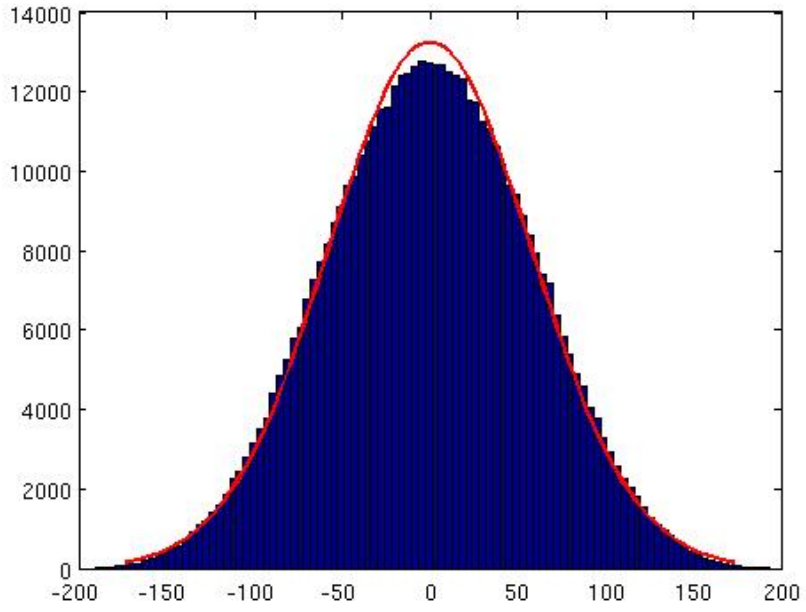

Figura 3.20: *Distribuzione gaussiana.*

Capitolo 4

Descrizione del sistema e dei dispositivi

L'architettura dell'emulatore proposta nel capitolo precedente viene realizzata con l'utilizzo di due categorie di dispositivi: quelli a calcolo spaziale e quelli a calcolo temporale. Con dispositivo a calcolo spaziale si intende una particolare soluzione hardware che permette di distribuire il carico computazionale su più risorse. In sostanza, con l'approccio a calcolo spaziale si può sfruttare un elevato grado di parallelismo delle risorse. Il dispositivo a calcolo spaziale utilizzato e l'FPGA (Field Programmable Gate Array).

Parlando di calcolo temporale si ci riferisce invece al tipo di elaborazione seriale sequenziale tipicamente offerta dai microprocessori.

In figura 6 possiamo osservare la struttura dell'emulatore realizzato ove vengono evidenziate le interconnessioni tra i dispositivi di calcolo utilizzati. L'emulatore viene interfacciato con un Personal Computer al fine di caricare tutti i parametri necessari all'emulazione ovvero la statistica di emissione temporale, quella energetica, l'immagine che deve essere emulata, la forma d'onda dell'impulso ed altri parametri accessori di controllo. La struttura in figura 6 è caratterizzata da una board primaria (Zedboard) ove sono collocati un dispositivo a calcolo spaziale FPGA ed un microprocessore con la relativa memoria primaria, e quattro board secondarie. Ogni board secondaria è interfacciata attraverso un bus parallelo sincrono a 16 bit di dato e 4 di indirizzo alla scheda master. Su di essa trovano posto un dispositivo FPGA

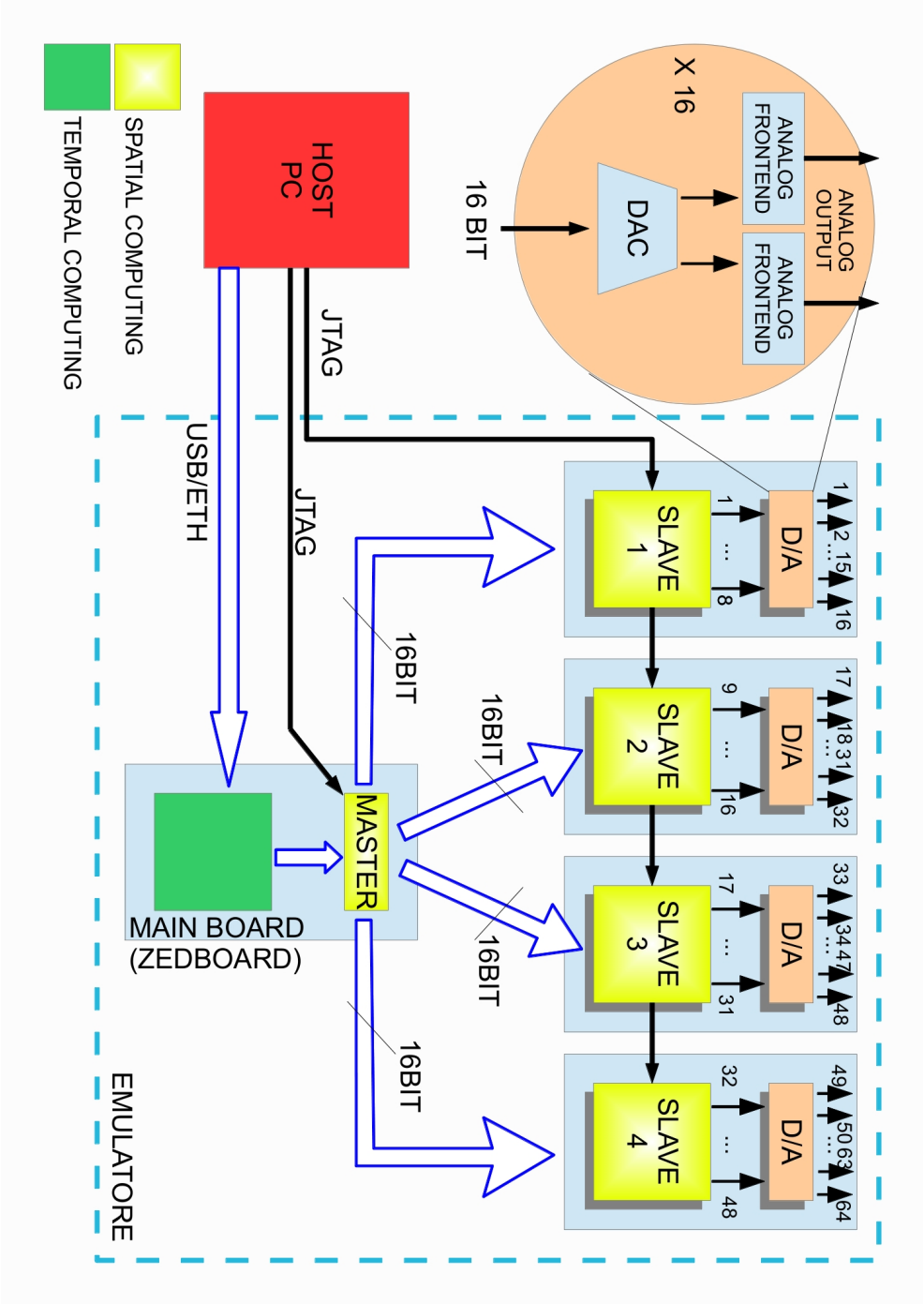

Figura 4.1: *Struttura schematica dell'emulatore.*

e 8 stadi di conversione digitale analogico interfacciati al dispositivo di calcolo attraverso bus a 16 bit. Ogni stadio di conversione digitale analogico include

un convertitore DAC e due stadi di condizionamento analogico; per questo ogni scheda secondaria è provvista di 16 uscite analogiche, ognuna delle quali rappresenta l'uscita di un canale di emulazione. L'emulatore è quindi in grado di generare mediante le quattro board secondarie segnali analogici per 64 detector.

4.1 Partizionamento dell'emulazione

Sulla struttura hardware descritta viene distribuita l'elaborazione dell'emulatore. Lo schema 4.2 mostra il partizionamento dell'emulazione adottato. Il processo di emulazione opera in tempo reale ed i dispositivi che realizzano i segnali emulati devono calcolare un gran numero di operazioni matematiche al secondo. Volendo fare una semplice stima del carico computazionale derivante dall'emulazione di 64 detector, imponendo che ogni canale debba essere alimentato da un flusso di parole digitali di circa 200M parole al secondo, otteniamo un rate di circa 25Gbyte/s. Come si può facilmente intuire questi dati non possono essere salvati in memoria né tantomeno possono essere prodotti da un microprocessore. L'emulazione richiede l'uso di un FPGA per generare i campioni che devono essere trasmessi ai convertitori DAC e trasformati in segnali di tensione. L'unica elaborazione che viene affidata ad un microprocessore è l'emulazione della posizione. Si è deciso di procedere in questa direzione in quanto la complessità e la quantità di memoria necessaria sono eccessive per una implementazione su FPGA, d'altro canto in questo caso non è richiesto il rate di campioni appena visto. Infatti per ogni forma d'onda l'emulatore della posizione deve fornire soli 64 valori, che come sappiamo sono i coefficienti che scalano le altezze dei 64 impulsi coerentemente con la posizione emulata. In questo caso il rate che ci si aspetta e di 1M conteggi/s.

Il dispositivo temporal computing e quello master spatial computing convivono in un unico dispositivo fisico, denominato Zynq. La comunicazione fra di essi è pertanto agevolata dal fatto che è possibile sfruttare bus interni di comunicazione.

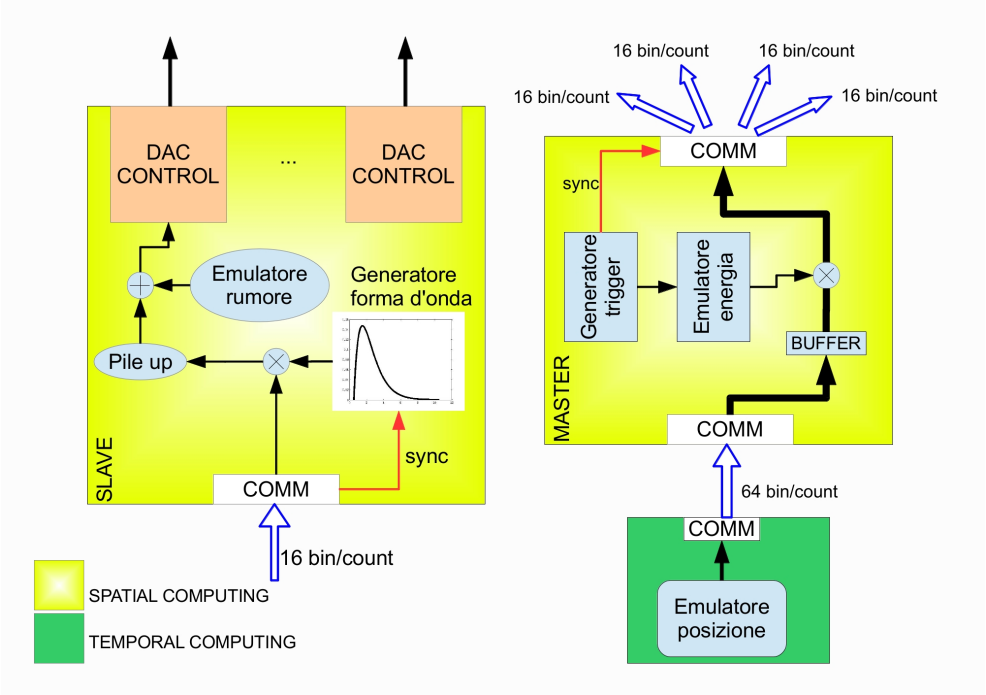

Figura 4.2: *Schema del partizionamento dell'emulazione.*

Su questi bus viene veicolato il flusso di dati prodotto dall'emulatore della posizione. Come possiamo vedere in figura, il bus è unidirezionale in quanto i valori di scalatura delle ampiezze degli impulsi vengono generati al rate medio di emissione senza la necessità di avere sincronismo con l'emulatore della statistica temporale.

I coefficienti dell'emulatore della posizione vengono moltiplicati, all'interno del dispositivo FPGA master, con quelli relativi al livello di energia. Il trigger, quindi, sincronizza le 4 schede slave ovvero stimola la generazione delle forme d'onda.

Nel seguito del capitolo analizzeremo le caratteristiche dei principali dispositivi utilizzati nella realizzazione dell'emulatore.

Il processore ARM è il dispositivo temporal computing scelto per l'implementazione dell'emulatore della posizione e convive quindi con una FPGA all'interno dello Zynq.

4.2 ARM

Con l'acronimo ARM (Advanced RISC Machine) non si intende un solo microprocessore, ma un'intera famiglia, dalle CPU più semplici con una sola pipeline a CPU molto complesse per applicazioni ad alte prestazioni. Oggi è probabilmente l'architettura più diffusa per sistemi embedded che vanno dai cellulari all'I-Pad a un grandissimo numero di applicazioni di ogni genere (per esempio il 78% dei telefoni cellulari usa processori ARM).

ARM come azienda non produce silicio ma "vende" proprietà intellettuale (IP) ovvero l'insieme delle istruzioni, la microarchitettura delle CPU, fino al progetto elettronico completo. Di norma, i processori ARM compaiono come "componenti" (detti core) di circuiti integrati più complessi ove tipicamente si trovano oltre alla CPU memorie, controllori di periferiche, dispositivi dedicati a specifici compiti quali elaborazione di segnale. Il primo prototipo nacque negli anni 80 a partire dall'architettura MIPS. Essa è sostanzialmente un'architettura di tipo *load store* che implica che tutte le istruzioni aritmetico logiche possono operare solo sui dati presenti nei registri interni della CPU, mentre per effettuare trasferimenti in memoria occorrono istruzioni specifiche. Le istruzioni aritmetico logiche specificano i registri dei due operandi sorgente e di quello di destinazione oltre ovviamente al codice operativo che determina il tipo di istruzione; questa caratteristica determina la struttura architetturale "a tre operandi". L'esecuzione di una istruzione non è contenuta all'interno di un singolo ciclo macchina ma viene suddivisa in fasi consecutive all'interno di una struttura ad esecuzione sequenziale denominata pipeline. La pipeline originale dell' ARM è costituita da tre blocchi; un blocco di lettura dell'istruzione (fetch), un blocco di decodifica (decode) dove la macchina interpreta il tipo di istruzione letta, ed un blocco di esecuzione (execute) dove a partire dai valori degli operandi sorgente viene prodotto il risultato dell'operazione che viene salvato nell'opportuno registro destinazione. La prima pipeline a tre stadi si è subito rivelata inadeguata ed è stata ben presto rimpiazzata da una nuova a cinque stadi; lettura, decodifica, esecuzione, accesso alla memoria primaria, scrittura nel registro destinazione.

L'evoluzione della pipeline ha accompagnato la modifica del bus interno alla CPU: dall'approccio Von Neumann che prevedeva un unico bus ed un'unica memoria si è giunti all'approccio Harvard con memoria e bus per dati e istruzioni separate. Ciò ha comportato un'esecuzione non bloccante nel caso, che comunemente avviene in qualsiasi architettura pipelined, di lettura di un' istruzione simultanea alla scrittura dati nei registri. All'architettura base vengono aggiunti uno o più coprocessori e i così detti percorsi di forwarding che consentono di risolvere problematiche dovute alla non disponibilità dei dati durante l'esecuzione: le istruzioni vengono lanciate nella pipeline una dopo l'altra; in linea di principio a regime ad ogni ciclo macchina si ha un'istruzione completata. Tuttavia se una istruzione produce un risultato che verrà utilizzato come sorgente per una istruzione subito successiva ad essa, nella pipeline a 5 stadi appena illustrata, tale valore viene scritto nei registri solo nell'ultimo stadio. Ciò implica che l'istruzione successiva non può essere eseguita subito dopo la precedente mancando un suo operando sorgente. I percorsi di forwarding trasportano all'indietro nel flusso di esecuzione della pipeline i dati così che ogni istruzione possa fluire senza blocchi.

Un coprocessore è un'architettura di calcolo che consente di eseguire specifiche operazioni matematico-logiche su specifici tipi di dati che il processore non è in grado di eseguire. Un esempio per l'architettura utilizzata è l'unità FP (Floating Point) che consente di effettuare operazioni in virgola mobile ad alta efficienza.

Il grande successo dell'architettura Harvard è sicuramente determinato dalla sua versatilità e dalla bassa dissipazione di potenza. E' importante sottolineare che da un punto di vista architetturale, ovvero nella capacità di organizzare le risorse e sfruttare le tecnologie di cui si dispone, è di norma più conveniente distribuire il carico computazionale al fine di migliorare le performance del sistema piuttosto che aumentarne la sua sofisticazione e velocità. Questo principio sta alla base del parallelismo offerto dai processori multicore. Sebbene tale approccio comporti difficoltà nella gestione della coerenza dei dati, ovvero la necessità che ciascun core abbia sempre a disposizione i dati aggiornati dagli altri, esso ha dimostrato la sua validità in fase di testing: un processore a due core @100Mhz offre migliori prestazioni,

soprattutto quando si deve eseguire codice in parallelo, di un singolo core @200Mhz. Questo non deve sorprendere, infatti, diversamente dal sistema dual core, se due processi devono essere eseguiti in parallelo su un processore single core è necessario effettuare ripetute volte il così detto context switching cioè riprogrammare il processore per fare eseguire alternativamente uno solo dei due processi. Nell'effettuare questa operazione si perde tempo utile ai fini dell'esecuzione vera e propria.

Una delle grosse problematiche associata alla gestione dei dati in un sistema multicore riguarda la coerenza della memoria. Al fine di aumentare il numero medio di istruzioni eseguito all'interno del ciclo macchina, una delle strategie adottabili, come avviene nella totalità degli odierni calcolatori, è quella di inserire livelli intermedi di memoria tra il processore e la memoria primaria. Tale tecnica prende il nome di gerarchia di memoria. Una gerarchia di memoria classica prevede l'inserimento di due o tre livelli intermedi di memoria con particolari caratteristiche. Siccome il processore lavora ad una frequenza molto alta esso richiederà altrettanto velocemente alla memoria i dati necessari per l'elaborazione. Se la memoria è lenta il processore dovrà attendere la disponibilità di questa prima di proseguire l'esecuzione perdendo tempo utile all'elaborazione. Per dare l'illusione al processore di poter comunicare con una memoria grande e veloce, che sono caratteristiche antagoniste nei sistemi di memoria, si utilizza una memoria piccola e veloce denominata *cache* collegata da un lato alla memoria primaria grande e lenta e dall'altra al processore. Poiché in un codice è frequente l'utilizzo ripetuto degli stessi dati (es. dei vettori) così come di istruzioni appartenenti allo stesso loop (while), la memoria cache riesce a soddisfare nella maggioranza dei casi le richieste del processore. Quando però si richiede un dato non presente in cache questa viene aggiornata interrogando la memoria primaria. Da quanto illustrato si evince che la memoria non contiene necessariamente in qualsiasi momento gli ultimi dati scritti dal processore che invece si ritrovano in cache. Le cose si fanno ancora più complicate in un sistema multicore dove tipicamente il primo livello di cache è riservato allo specifico core ed il secondo condiviso. Vengono allora introdotte le politiche di *cache coherence* al fine di ottenere sempre valori corretti alla fine di una certa sezione dell'elaborazione.

4.3 Zynq 7000

La famiglia Zynq-7000 è basata sull'architettura Xilinx all programmable SOC (System On Chip). Questi prodotti integrano un processore dual core Cortex-A9 ARM ricco di caratteristiche che costituisce la parte PS (Programmable System) del sistema e una logica programmabile PL sviluppata con tecnologia a 28nm in un singolo dispositivo. Le CPU ARM Cortex-A9 rappresentano il cuore del PS che tra le altre cose include una memoria on-chip, interfacce di memoria esterna, ed un ricco set di periferiche di connettività.

Lo Zynq offre la flessibilità e scalabilità di una FPGA esibendo le performance, la potenza di calcolo, e la semplicità nell'utilizzo tipiche di un ASIC (Application Specific Integrated Circuit). Mentre ogni dispositivo della famiglia contiene lo stesso PS, la logica programmabile e le risorse di input/output possono variare garantendo così la possibilità di soddisfare un ampio ventaglio di applicazioni; dall'automotive alla connettività LTE, all'imaging medico diagnostico. L'architettura consente lo sviluppo di logica programmabile nella PL e di software personalizzato nel PS ottenendo risultati che le due componenti singolarmente non potrebbero dare a causa di risorse di comunicazione limitate, latenze ed eccessivo consumo di potenza. La comunicazione tra la due parti del sistema, flessibile ed efficace, si basa sul bus AXI. L'inclusione di un processore performante permette il supporto di un sistema operativo ad alto livello quale, ad esempio Linux. Poiché in fase di accensione è sempre il PS ad essere abilitato per primo si ha un approccio all'esecuzione incentrato sul software. La PL è dunque vista dal software come una sua naturale estensione.

In figura 5.2 possiamo vedere la tipica struttura dell'architettura. La parte inferiore è occupata dalla PL che tra le altre cose include il supporto all'AXI bus attraverso le "general purpose ports" e le "high performance ports".

In alto a sinistra individuiamo il blocco Application Processor Unit (APU) che include:

- Processore dual core ARM Cortex-A9 MPCores con caratteristiche per ogni core di 2.5 MIPS/Mhz (Milion Instruction Per Second), clock

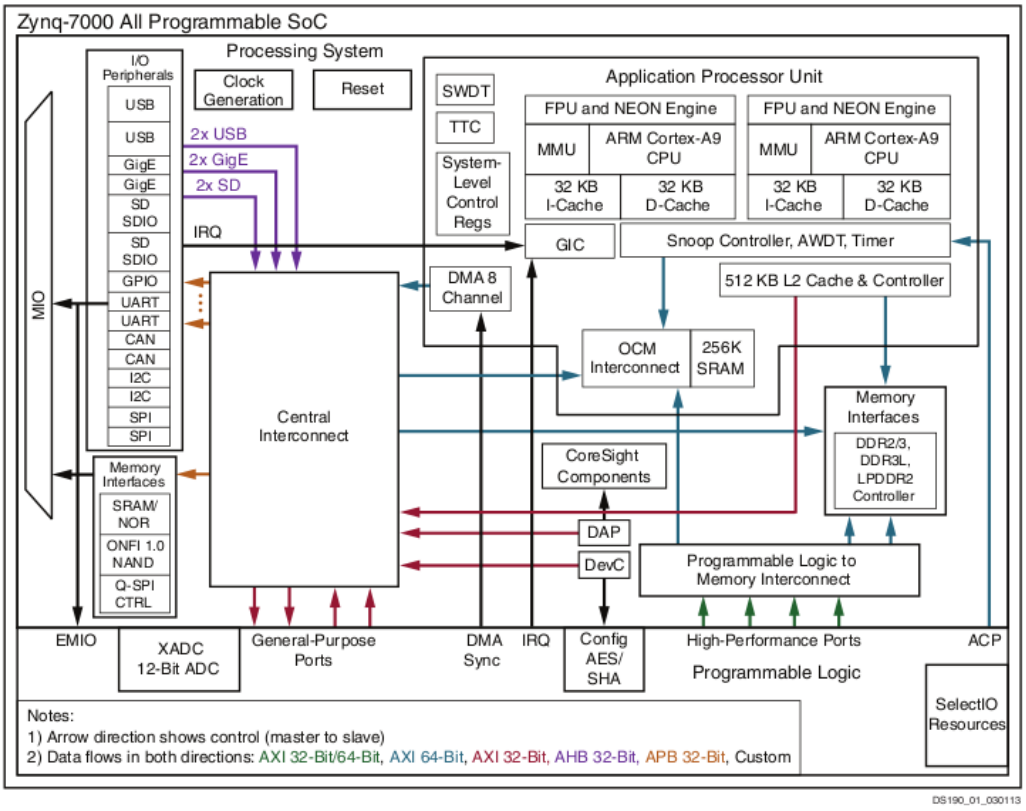

Figura 4.3: *Struttura Zynq.*

@667Mhz, unità in virgola mobile con una capacità di calcolo di 2.0 MFLOPS/MHz (Milion Floating Point Operation Per Second), supporto per approccio architetturale SIMD (Single Instruction Multiple Data, ovvero la possibilità che entrambi i core seguano le stesso flusso di esecuzione su due diversi flussi di dati), 32kB di cache di primo livello set associativa a quattro vie, Memory Managment Unit (MMU).

- Controllore ACP (Accelerator Coherency Port) per accessi cache coherent da parte della logica programmabile
- 512kB di cache di secondo livello unificata set associativa ad otto vie.
- Ram on chip OCM (On Chip Memory) di 256kB

Un ruolo importante è ricoperto dal blocco Central Interconnect. Attraverso di esso infatti il processore può comunicare con tutte le periferiche di cui il

sistema è dotato. Tra le più importanti vi sono il modulo di comunicazione seriale UART, l' USB, l'interfaccia di rete GigabitEthernet e l' SPI. Anche Il controllore di DMA integrato nell'Application Processor Unit si interfaccia con questo blocco di comunicazione. Sulla sinistra notiamo un blocco MIO (Multiplexed Input Output) la cui funzione è quella di multiplexare l'accesso dalle periferiche del PS e dalle interfacce di memoria (SRAM, NAND) ai pin del PS stesso, come impostato attraverso opportuni registri di configurazione. Sono disponibili 54 pins per l'insieme delle periferiche di input output e per le interfacce di memoria sopra citate all'interno del PS. E' importante sottolineare che il processore con architettura a 32bit è in grado di risolvere uno spazio di indirizzamento $2^{32} = 4GB$ all'interno del quale vengono mappate tutte le periferiche ad esso connesse (approccio memory mapped).

Il controllore di memoria DDR (Double Data Rate) multi protocollare, sulla destra in figura 5.2, può essere configurato per indirizzare 1Gword di parole a 16 o 32 bit utilizzando un'unica classe di memorie DRAM a 8, 16 o 32 bit. LA velocità di comunicazione può raggiungere i 1333Mb/s se è supportata dai banchi di memoria la modalità DDR3. Il controllore di memoria permette un accesso multiporta rendendo possibile la condivisione della memoria centrale tra il processore e la logica programmabile. Sono infatti presenti 4 porte ad alta velocità, interfacciabili con bus AXI, con le seguenti caratteristiche:

- Una porta a 64 bit, collegata al controllore di cache di secondo livello, è dedicata al processore ARM e può essere configurata per avere bassa latenza.
- Due porte a 64 bit sono riservate per l'accesso da parte della logica programmabile PL.
- Una porta a 64 bit è condivisa per l'accesso attraverso il Central Interconnect.

Lo Zynq usa il processo di boot multi stadio che supporta sia una modalità sicura che non. Il PS è il master del processo di avvio e configurazione. Per avere boot sicuro la PL deve essere abilitata per usufruire del blocco di

sicurezza che fornisce una crittografia AES a 256 bit. Una volta effettuato il reset o subito dopo l'accensione viene letto lo stato dei pin di modalità per determinare il dispositivo di avvio primario. Esso può essere un chip di memoria NAND o NOR, o una scheda SD. L'interfaccia JTAG permette il boot per scopi di debugging ed è da utilizzarsi ad esempio in prototipazione. Un solo core del processore copia il boot loader di primo stadio (FSBL) dal dispositivo di boot primario alla memoria interna OCM e ne esegue il codice. Tipicamente l' FSBL carica un applicativo eseguibile in modalità stand-alone o un secondo boot loader più sofisticato quale ad esempio U boot. Nel caso di utilizzo del kernel linux, come vedremo nel seguito, sarà compito di questo secondo boot loader inizializzare l'esecuzione del sistema operativo vero e proprio.

Il sistema possiede due porte JTAG che possono essere concatenate o usate separatamente. Nel caso di concatenamento viene usata un'unica porta per il caricamento del codice eseguibile da parte del processore ARM, per le operazioni di controllo durante l'esecuzione, e per la configurazione ed il debug della logica programmabile.

Alcune importanti caratteristiche relative alle funzionalità di input-output della logica programmabile includono:

- Alte performance con supporto alla trasmissione dati DDR3 @ 1866Mb/s
- Capacità di decoupling ad alta frequenza all'interno del package del dispositivo per avere caratteristiche avanzate di signal integrity.
- Impedenza controllata digitalmente che può essere impostata in modalità 3-state per comunicazione ad alta velocità e basso consumo di potenza.

Il numero di pin di input output varia in funzione del modello considerato; in particolare nel modello utilizzato XC7Z020 (package CLG484) si hanno 200 pin ad elevata dinamica con tensione tra 1,2V a 3,3V. Ogni pin può soddisfare diversi standard di I/O; ad eccezione dei pin di alimentazione e di pochi altri pin di configurazione, tutti i pin della PL hanno le stesse capacità, vincolate solo da certe regole relative alla divisione in banchi. Infatti i pin sono

organizzati in banchi di 50 pin ognuno. Ogni banco ha un'unica tensione di uscita VCCO che tra l'altro alimenta anche alcuni buffer di ingresso. Alcuni di essi, in modalità single-ended, richiedono una tensione di riferimento Vref generata internamente o proveniente dall'esterno del banco. Alcune capacità con resistenza serie equivalente ESR controllata sono montati sul substrato del package per ottimizzare la signal integrity in caso di commutazione simultanea delle uscite.

Tutti gli ingressi e le uscite relative alla logica programmabile possono essere configurati in modalità DDR. Qualsiasi ingresso e alcune uscite possono essere ritardati fino a 32 unità di 78ps o 52ps ognuna. Il numero di blocchi di ritardo può essere impostato run-time.

4.3.1 AXI bus

Lo Zynq prevede la comunicazione tra logica programmabile e sistema programmabile (processore) attraverso un bus specifico; l'AXI bus. AXI (Advanced eXtensible Interface) fa parte di AMBA (Advanced Microcontroller Bus Architecture), una famiglia di bus per microcontrollori introdotta nel 1996. La prima versione dell' AXI venne inclusa nella versione AMBA 3 rilasciata nel 2003. AMBA 4, rilasciato nel 2010, include la seconda versione dell' AXI; AXI4. Esistono tre tipi di interfacce basate su AXI:

- AXI4 utilizzato ove è necessario avere un approccio memory mapped ed elevate performance.
- AXI4-Lite utilizzato per comunicazioni semplici a basso throughput ma sempre basate su approccio memory mapped.
- AXI4-Strem per la comunicazione streaming ad alta velocità.

Le specifiche AXI impongono delle regole relative alla comunicazione tra un'interfaccia master e una slave collegate attraverso una struttura denominata blocco di interconnessione. L'approccio memory mapped impone che ogni transazione debba coinvolgere uno specifico indirizzo di destinazione all'interno di uno spazio di indirizzamento. Il master del bus ha il compito di inizializzare la transazione di lettura o scrittura dello slave. La struttura del

bus permette la trasmissione e la ricezione simultanea dei dati che possono essere raggruppati in pacchetti detti burst. Questa possibilità non è però offerta dalla modalità AXI4-Lite caratterizzata da una singolo trasferimento per transazione.

In effetti la modalità Lite è tipicamente utilizzata per impostare opportuni registri di configurazione mentre la modalità normale è impiegata nella trasmissione vera e propria.

Nel protocollo AXI4-Stream viene definito un unico canale per la trasmissione continua dei dati; infatti in questa modalità il numero di dati componenti un burst è illimitato.

Come illustrato lo Zynq possiede due porte di interconnessione tra PS e PL. La porta a bassa velocità supporta solo l'AXI4-Lite mentre quella ad elevate performance supporta l'AXI4. Il processore può effettuare bus mastering, ovvero l'arbitraggio della comunicazione sul bus, solo lato AXI4-Lite. Nell'architettura Zynq è dunque necessario utilizzare una soluzione che sfrutti le porte ad alta velocità, facendo bus mastering lato PL, per poter effettuare trasferimenti veloci.

4.3.2 DMA e CDMA

Un controllore DMA (Direct Memory Access) è un dispositivo capace di effettuare in autonomia trasferimenti di dati all'interno di un'architettura a processore. In un sistema di calcolo si ha spesso la necessità di trasferire una certa mole di dati da periferiche alla memoria centrale. In un sistema di elaborazione video ad esempio è necessario salvare in memoria i frame prodotti dalla periferica di acquisizione video prima di fare qualsiasi tipo di elaborazione su di essi. In assenza di controllore DMA la CPU è costretta a spostare parola dopo parola la successione di immagini che compongono il video da un buffer locale alla periferica di acquisizione alla memoria centrale. Ciò comporta un costo computazionale elevato per il processore che è costretto per lungo tempo a trasferire i dati piuttosto che effettuare il rendering video vero e proprio. Con l'ausilio del DMA il processore viene alleggerito dell'onere del trasferimento poiché il controllore può effettuare bus mastering diver-

samente da tutte le altre periferiche (slave) collegate al bus. Il processore periodicamente programma il controllore di DMA tramite opportuni registri ed avviato il trasferimento desiderato continua l'esecuzione. Visto l'indubbio vantaggio che tale meccanismo offre si sono avute successive evoluzioni come l'EDMA (Enhanced DMA) che richiedono sempre minor carico per la CPU.

Il CDMA (Central DMA) è una proprietà intellettuale sviluppata da Xilinx consistente in un DMA implementato in logica programmabile su FPGA. Il CDMA si interfaccia tramite AXI. In particolare i suoi registri di configurazione sono accessibili attraverso AXI4-Lite mentre l'interfaccia in grado di effettuare bus mastering viene collegata mediante AXI. Il CDMA supporta due modalità DMA: quella canonica, nella quale si specifica l'indirizzo sorgente dei dati, quello di destinazione e il numero di parole da trasferire, e la modalità scatter che da la possibilità di preprogrammare una coda di trasferimenti che verranno finalizzati in sequenza. Il trasferimento dei dati può essere compiuto in due modi distinti:

- Dopo aver abilitato il trasferimento il CDMA interroga la periferica corrispondente al primo indirizzo programmato e salva uno per uno tutti i dati in un buffer locale, dopo di che scrive tutte le locazioni di destinazione. Ogni operazione di scrittura o lettura richiede un certo intervallo di tempo T e quindi il tempo totale necessario per trasferire i dati è $T^{tot} = T * 2 * N_{data}$.

- Il CDMA legge il primo dato sorgente e lo scrive nella destinazione. Il processo si ripete per tutti i dati. Se una operazione di lettura + scrittura richiede un tempo $2T$ allora T^{tot}non cambia rispetto a prima.

Le due modalità sono del tutto equivalenti eccetto per il fatto che nel primo modo si possono identificare due lunghi intervalli di tempo; infatti la periferica sorgente continua a trasmettere dati al CDMA mentre la periferica destinazione rimane inattiva. Nel secondo i ruoli si invertono; il CDMA continua a programmare la periferica di destinazione mentre la sorgente è inattiva. Questa circostanza permette, soprattutto in situazioni di elevato traffico, che vi siano periodi di inattività delle periferiche utilizzabili per altri scopi come vedremo nel seguito.

4.3.3 Zedboard

Lo Zynq è installato sulla scheda di prototipazione *Zedboard* visibile in figura *4.4*. La board contiene tutto il necessario per realizzare un sistema a piattaforma Linux, Android o Windows. Le caratteristiche principali sono:

- 512 MB di memoria DDR3
- 1 porta USB master e 1 porta USB OTG
- display OLED con risoluzione di 128x32 pixel
- interfaccia Gigabit Ethernet con connettore RJ-45
- connettore FMC
- porta VGA ed HDMI
- switch e LED
- JTAG per la programmazione dello Zynq e interfaccia UART per console seriale
- lettore scheda SD

La Zedboard include due chip Micron MT41J128M16HA-15E:D di memoria RAM DDR3 per creare una interfaccia a 32 bit. La memoria RAM è collegata al controllore di memoria all'interno del PS dello Zynq. Il controllore DDR multi protocollo è configurato per effettuare un accesso a 32 bit su 512MB di spazio di indirizzamento. L'interfaccia supporta una frequenza di 533Mhz (1066Mbs).

Il connettore video collegato ad un DAC a resistenze pesate, realizzato sulla parte FPGA dello Zynq, in congiunzione alla possibilità di collegare una tastiera alla porta USB master permettono l'utilizzo della scheda come work station in fase di testing.

Ruolo importante è rivestito dal connettore FMC visibile sulla destra di figura 4.4. L'FMC è costituito da 68 porte di input-output single ended ad impedenza controllata di 50Ω che possono essere configurate come 34

coppie differenziali. L'interfaccia FMC si estende sui banchi 34 e 35 associati alla PL. Per soddisfare le specifiche dello standard FMC questi banchi sono alimentati con una tensione regolabile dall'utente attraverso un jumper. Le tensioni selezionabili sono 1.8V di default, 2,5V e 3,3V.

Attraverso i jumper, posizionati nella parte superiore della scheda in prossimità del connettore FMC, è possibile selezionare la modalità di boot. La scheda supporta il boot da JTAG o da SD ove si è precedentemente caricato il boot loader di secondo livello ed il sistema operativo.

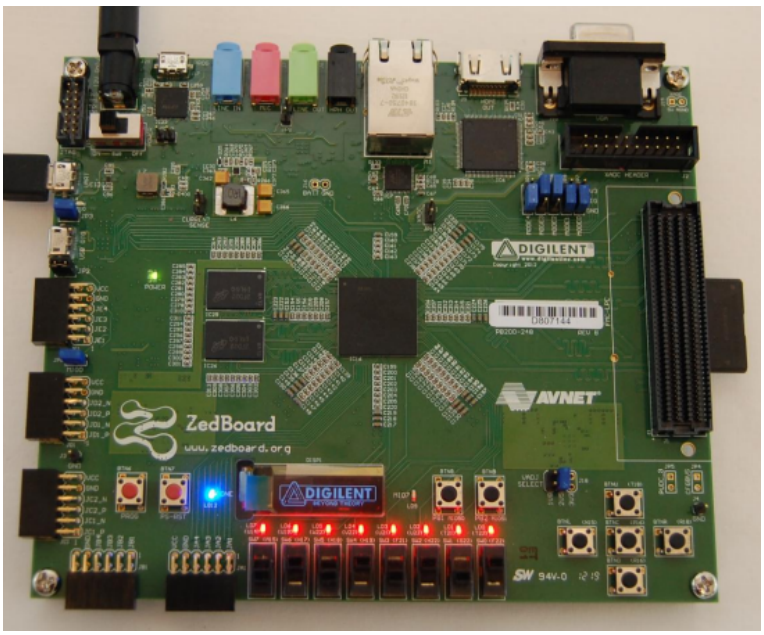

Figura 4.4: *Scheda di prototipazione Zedboard.*

4.4 FPGA

Un Field Programmable Gate Array, solitamente abbreviato in FPGA, è un circuito integrato digitale programmabile via software con il quale è possibile sintetizzare complesse logiche combinatorie e sequenziali.

Sono elementi che presentano caratteristiche intermedie rispetto ai dispositivi ASIC (Application Specific Integrated Circuit) e a quelli con architettura PAL (Programmable Array Logic). L'uso di componenti FPGA comporta

alcuni vantaggi rispetto agli ASIC: si tratta infatti di dispositivi standard la cui funzionalità da implementare non viene impostata dal produttore che quindi può produrre su larga scala a basso prezzo. La loro genericità li rende adatti a un gran numero di applicazioni come consumer, comunicazioni ed automotive. Essi sono programmati direttamente dall'utente finale, consentendo la diminuzione dei tempi di progettazione, di verifica mediante simulazioni e di prova sul campo dell'applicazione. Il grande vantaggio rispetto agli ASIC è che permettono di apportare eventuali modifiche o correggere errori semplicemente riprogrammando il dispositivo in qualsiasi momento. Per questo motivo sono utilizzati ampiamente nelle fasi di prototipazione, in quanto eventuali errori possono essere risolti semplicemente riconfigurando il dispositivo. Di contro, per applicazioni su grandi numeri sono antieconomici perché il prezzo unitario del dispositivo è superiore a quello degli ASIC (che di converso hanno elevati costi di progettazione). Usualmente vengono programmati con linguaggi come il Verilog o il VHDL.

La LUT (Look Up Table) è sicuramente il cuore del funzionamento di un FPGA. E' noto che qualsiasi funzione logica può essere implementata attraverso una opportuna combinazione di porte logiche AND, OR e NOT. Esiste però un modo alternativo che si basa sulla LUT. Supponiamo di dover implementare una certa funzione logica a 4 ingressi $z = f(\alpha, \beta, \gamma, \delta)$ e di calcolare per ogni possibile combinazione di questi i corrispondenti valori dell'uscita z. Se si utilizza una memoria di dimensione $2^4 = 16$ bit con ogni cella di dimensione 1 bit è possibile salvare al suo interno tutti i possibili valori di z. Un modo intelligente per memorizzarli è il seguente. Associamo ad ogni combinazione $\alpha, \beta, \gamma, \delta$ un indirizzo della memoria. Nella cella indirizzata verrà memorizzato il risultato dell'operazione logica dato da quella specifica sequenza degli ingressi. Così facendo il bit di uscita della memoria rappresenta z. Questo modo di sintetizzare una qualsiasi funzione logica a 4 ingressi si può estendere ad n ingressi ed ha il grosso vantaggio di avere sempre la stessa implementazione per qualsiasi funzione da sintetizzare. Tra l'altro utilizzando una memoria riprogrammabile si toglie il vincolo di dover sapere a priori la funzionalità di un certo blocco implementativo.

La complessità delle FPGA è nel tempo aumentata. In linea di massima

possiamo pensare l'architettura come divisa in blocchi funzionali come in figura 4.5 che una volta programmati dall'utente espleteranno la funzione stabilita.

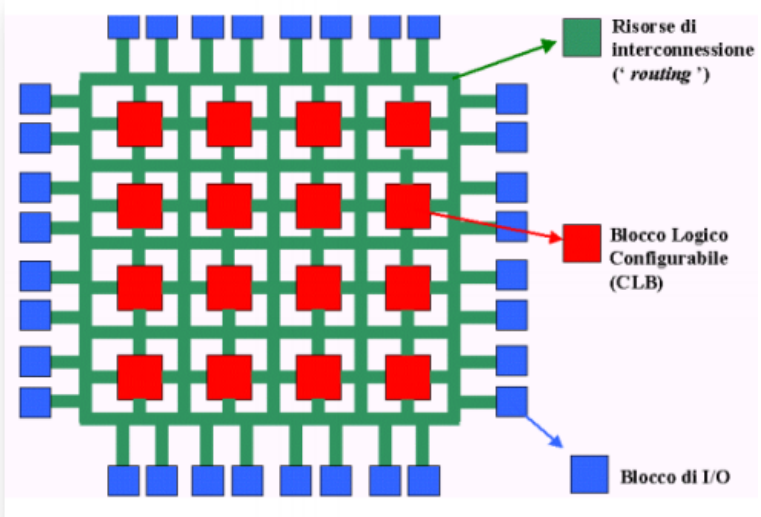

Figura 4.5: *Struttura FPGA.*

In figura notiamo la presenza di tre blocchi distinti:

- I blocchi di I/O sono pensati per effettuare con efficienza tutte le operazione relative alla comunicazione del sistema con l'esterno, non a caso tali blocchi sono localizzati sul perimetro del dispositivo.
- I blocchi di logica configurabile effettuano l'elaborazione dei segnali; al loro interno troviamo strutture di LUT.
- Le strutture di interconnessione sono tutti i possibili collegamenti tra i vari blocchi. Poiché il sistema è completamente programmabile, sia per quanto riguarda le funzionalità dei due tipi di blocchi precedenti che le loro relative interconnessioni, è necessario avere una certa ridondanza di collegamenti. Un certo blocco può avere la necessità di comunicare con qualsiasi altro del sistema.

A questa struttura elementare si sono aggiunte risorse specifiche come ad esempio delle memorie RAM denominate BlockRAM; dei DigitalSignalPro-

cessing Block, veri e propri dispositivi DSP per l'elaborazione dei segnali; dei Microblaze Block, un semplice microprocessore, oppure dei moltiplicatori.

L' FPGA utilizzata appartenente alla serie Kintex7 prodotta dalla Xilinx è il modello XC7K325T in package ffg676. Le caratteristiche principali sono: 326000 celle logiche e 16Mbit di memoria BRAM , 10 banchi di input output per un totale di 500 pin di cui 250 ad elevata dinamica (da 1.2V fino a 3.3V) e 150 ad alta velocità con un range massimo in tensione configurabile tra 1.2V e 1.8V.

Le BlockRAM a doppia porta di accesso, sono strutturate in moduli di 36Kbit. Ogni BRAM può essere suddivisa in due blocchi completamente indipendenti di 16Kb ognuno suddivisibili a loro volta a discrezione dell'utente. E' anche possibile collegare in cascata due moduli per avere 64K di memoria senza impiegare logica supplementare.

Le uscite single-ended utilizzano una struttura push-pull CMOS che può trovarsi in stato alto, basso o ad alta impedenza. La lettura del valore dell'uscita è sempre attiva ma è ignorata quando il pin è configurato come uscita. Nel caso due pin vengano configurati come coppia d'ingresso differenziale è possibile utilizzare un resistore interno di terminazione di 100Ω.

Come per la PL dello Zynq, tutti gli ingressi e le uscite possono essere configurati in modalità Double Data Rate (DDR). Qualsiasi ingresso e alcune uscite possono essere ritardati fino a 32 unità di 78ps o 52ps ognuna. Il numero di blocchi di ritardo può essere impostato run-time.

4.5 DAC

Di sicuro un punto chiave nel processo di emulazione è la conversione digitale analogica dei segnali prodotti nell'elaborazione. Siccome quest'operazione non corrisponde ad una reale fase della rivelazione dei raggi gamma, è necessario che sia il più possibile trasparente in termini di errore aggiunto. Il convertitore digitale analogico (DAC) effettua due funzioni basilari: la transcodifica, che converte l'ingresso digitale in un segnale analogico equivalente, e la ricostruzione. La ricostruzione è necessaria per rimuovere le componenti

ad alta frequenza di un segnale analogico a dati campionati ed è realizzata in due fasi: un processo di *sample and hold* seguito da un filtraggio passa-basso.

Concettualmente la transcodifica genera una sequenza di impulsi, le cui ampiezze sono rappresentazioni analogiche di codici digitali. Successivamente la ricostruzione converte la sequenza di impulsi in un segnale tempo continuo. Normalmente un unico circuito effettua sia la transcodifica che il sample and hold mentre un filtraggio esterno addolcisce le forme d'onda a gradinata.

4.5.1 Non idealità dei DAC

Il segnale in uscita sintetizzato dal DAC presenta sempre degli errori più o meno gravi dipendenti dalla realizzazione del dispositivo. Tali errori vengono detti statici se il loro effetto si presenta in uscita in condizioni di regime, dinamici se, invece, si presentano sulle variazioni dell'uscita.

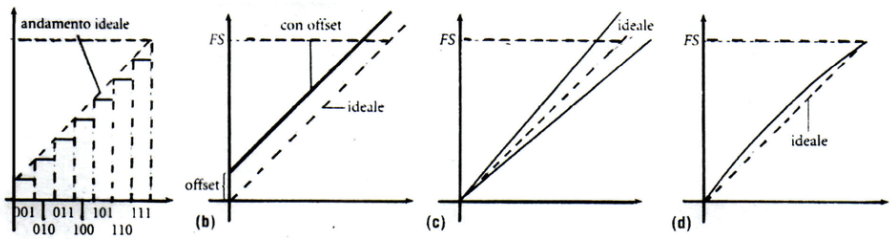

Figura 4.6: *Errori statici per i DAC.*

La figura 4.6.(a) descrive l'andamento ideale dell'uscita di un DAC a tre bit (scelta dettata da ragioni di comodità grafica) corrispondente a tutte le configurazioni d'ingresso crescenti da 000 a 111.

Unendo i vertici degli scalini dell'uscita nel modo indicato in figura, si ottiene una linea retta (straight line) che definisce l'andamento ideale desiderato per l'uscita di qualunque DAC. Tale andamento viene assunto come riferimento. Il comportamento reale di un DAC è valutato in termini di scostamento dal funzionamento ideale. Si individuano le seguenti principali tipologie di errore:

- **Offset.** Come mostrato in figura 4.6.(b), l'offset coincide con la tensione rilevabile all'uscita di un DAC quando la configurazione di ingresso

è composta solo da zeri. L'offset tende a mantenersi costante anche in corrispondenza di tutte le altre combinazioni d'ingresso. Pertanto, l'andamento dell'uscita soggetta a offset è rappresentato da una linea parallela a quella ideale e, in genere traslata verso l'alto di un valore pari a quello dell'offset stesso. L'offset è provocato dall'insieme delle tensioni continue generate dal funzionamento reale della circuiteria interna del DAC. Per questo motivo molti DAC integrati sono dotati di piedini di compensazione dell'offset.

- **Guadagno.** Lo scostamento dell'uscita effettiva rispetto a quella ideale si manifesta anche con una pendenza diversa da quella attesa. Come mostrato in figura 4.6.(c), l'uscita di un DAC può assumere una pendenza superiore o inferiore a quella ideale. Per tale ragione questo tipo di scostamento viene chiamato errore di guadagno (spesso indicato nei data sheet con il semplice termine di gain). È possibile compensare l'offset e il guadagno seguendo in modo preciso le indicazioni fornite dal data sheet di ciascun DAC. In genere, prima si annulla l'offset mediante una parola di ingresso composta solo da zeri e poi si regola il guadagno applicando in ingresso una parola di soli 1 e agendo in modo tale da ottenere una uscita corrispondente al valore di Fondo Scala (FS).

- **Linearità**. Come mostrato in figura 4.6.(d), l'andamento reale dell'uscita può assumere un andamento non lineare, cioè diverso da quello ideale (senza offset ed errori di guadagno). La linearità (Integral Non Linearity, INL) è definita come il massimo valore della deviazione presentata dall'andamento reale dell'uscita rispetto a quello ideale. La linearità viene misurata in percentuale del valore di Fondo Scala oppure in frazioni di 1 LSB. La linearità è normalmente contenuta entro $\pm$ (1/2) LSB.

- **Linearità differenziale**. Quando l'ingresso digitale commuta da una configurazione a quella successiva l'uscita dovrebbe aumentare di un gradino pari a un quanto Q e ciò dovrebbe verificarsi per qualunque commutazione dell'ingresso da una configurazione a quella successiva.

La linearità differenziale (Differential Non Linearity, DNL) misura la deviazione a cui è soggetta la variazione di un gradino dell'uscita reale rispetto al gradino dell'uscita ideale. Questo parametro viene misurato in sottomultipli di 1 LSB e normalmente assume un valore pari a ± (1/2) LSB.

- **Monotonicità.** Se la linearità differenziale supera il valore di 1 LSB allora l'uscita può aumentare (o diminuire) di un gradino di ampiezza superiore a Q. Ciò significa che quando l'ingresso passa da una configurazione a quella successiva l'uscita può aumentare (o diminuire) di un valore superiore a Q. In questi casi l'uscita assume un valore uguale in corrispondenza di diverse combinazioni di ingresso e, quindi, non è monotona.

Tra gli errori dinamici troviamo invece:

- **Tempo d'assestamento**. Si vorrebbe che l'uscita commutasse istantaneamente in seguito a una qualsiasi variazione della parola digitale di ingresso; nella realtà ciò non si verifica mai. Il tempo di assestamento (settling time) è l'intervallo di tempo necessario all'uscita di un DAC per raggiungere e mantenersi all'interno di una frazione (normalmente ± (1/2) LSB) del valore finale, in corrispondenza di una commutazione dell'ingresso. Il valore di questo parametro cambia al variare della commutazione dell'ingresso. Tipicamente le commutazioni considerate sono quelle relative a 1 un bit meno significativo (LSB), 1 bit più significativo (MSB) e al Fondo Scala.

- **Glitch**. Una qualunque commutazione dell'ingresso di un DAC dovrebbe generare la corrispondente commutazione dell'uscita in modo sicuro. Spesso capita che in corrispondenza di una commutazione dell'ingresso il DAC genera, seppur per un breve intervallo di tempo, una tensione d'uscita diversa da quella attesa. In generale, si parla di commutazioni principali (major transitions) e di commutazioni secondarie (minor transitions). La più importante commutazione si verifica quando l'ingresso del DAC vede una variazione del suo bit più significativo

(per esempio la transizione 011->100). In corrispondenza delle commutazioni principali, il DAC può fornire in uscita anche una tensione nulla per un breve intervallo di tempo (o in generale una tensione diversa da quella attesa) prima di raggiungere il valore corretto dell'uscita stessa. Questi transitori producono un andamento dell'uscita impulsivo. Tali transitori vengono chiamati glitch e possono assumere ampiezze di valore molto diverso. I glitch si possono verificare, in misura più contenuta, anche in corrispondenza delle commutazioni secondarie. La presenza dei glitch deve essere eliminata e ciò si ottiene mediante i cosiddetti circuiti deglitcher, che possono anche essere incorporati nello stesso chip del DAC. Normalmente i deglitcher sono realizzati mediante moduli Sample & Hold veloci. Il modulo campionatore mantiene costante la tensione di uscita fino a quando non si è generato il valore aggiornato corretto. In questo modo il valore dell'uscita, che ora corrisponde all'uscita dei deglitcher, risulta priva dei glitch. Ovviamente il funzionamento dei deglitcher deve essere opportunamente sincronizzato con quello del DAC.

- **Distorsione**. Osservando lo spettro di una forma d'onda ricostruita da un DAC a partire da un flusso di parole digitali noteremo la presenza di componenti armoniche non previste a contenuto di potenza relativamente alto. Queste componenti spurie derivano da distorsione e rumore. La distorsione può essere specificata in termini di distorsione armonica, SFDR (Spurious Free Dynamic Range) o di distorsione di intermodulazione tipicamente espresse in unità logaritmiche dBc. La distorsione armonica è definita come il rapporto tra le armoniche spurie e la fondamentale quando viene ricostruita una semplice sinusoide. L'SFDR è il rapporto della spuria con maggior potenza rispetto alla fondamentale. I glitch, di cui sopra, producono armoniche sia all'interno che all'esterno della banda del segnale.

- **SNR.** Il rapporto segnale rumore (SNR: Signal to Noise Ratio) viene espresso in dB e fornisce un'importante indicazione rispetto all'incidenza del rumore sul segnale utile. Il SNR dipende dalla risoluzione del

convertitore impiegato, cioè dal numero n di bit. Si dimostra che il valore di tale parametro nel caso di segnale sinusoidale è fornito dalla seguente relazione: SNR = (6,02n+1,76)dB

4.5.2 AD9747

AD9747 è prodotto dalla Analog Devices e integra in un solo chip due DAC con uscita differenziale in corrente e parole digitali di ingresso a 16bit. Il segnale di clock viene fornito attraverso una coppia di segnali differenziali. La scelta di tale dispositivo è stata dettata dall'esigenza di dover garantire ottime performance di linearità ad elevata frequenza di uscita (250Msample/s) pur mantenendo basso il costo del sistema. Viene garantito un errore di offset inferiore allo 0.001%, ed un errore di guadagno del 2% rispetto al fondo scala, DNL e INL tipiche rispettivamente di 2 e 4 bit meno significativi, distorsione di intermodulazione ed SFDR rispettivamente di 80dBc e 70dBc con frequenza dell'armonica in uscita di 70MHz.

Capitolo 5

Firmware

In questo capitolo si analizzerà il software realizzato per l'emulazione della posizione. Il software in esecuzione sul processore ARM dello Zynq assolve due compiti fondamentali che vengono realizzati attraverso l'impiego del sistema operativo Linux:

- Generazione valori di emulazione
- Trasmissione dei valori al dispositivo FPGA master

La struttura del software è visibile in figura 5.1.

Il codice primario, i cui dettagli implementativi analizzeremo nel seguito, genera i valori di modulazione degli impulsi relativi alla posizione di incidenza. Un device driver si occupa della comunicazione del software con l'hardware ovvero con la parte FPGA dello Zynq ove viene implementato un buffer di accumulo dei dati di emulazione: questa struttura prende il nome di deliverer. Il buffer è lo stesso presente in figura 4.2.

5.1 Configurazione Zynq

Una delle operazioni preliminari, necessarie per il buon funzionamento della Zedboard, consiste nella configurazione delle sue risorse. La scheda, per mezzo dell'ausilio di tool forniti dal costruttore, offre la possibilità di regolare molte impostazioni a seconda delle esigenze dello sviluppatore come ad

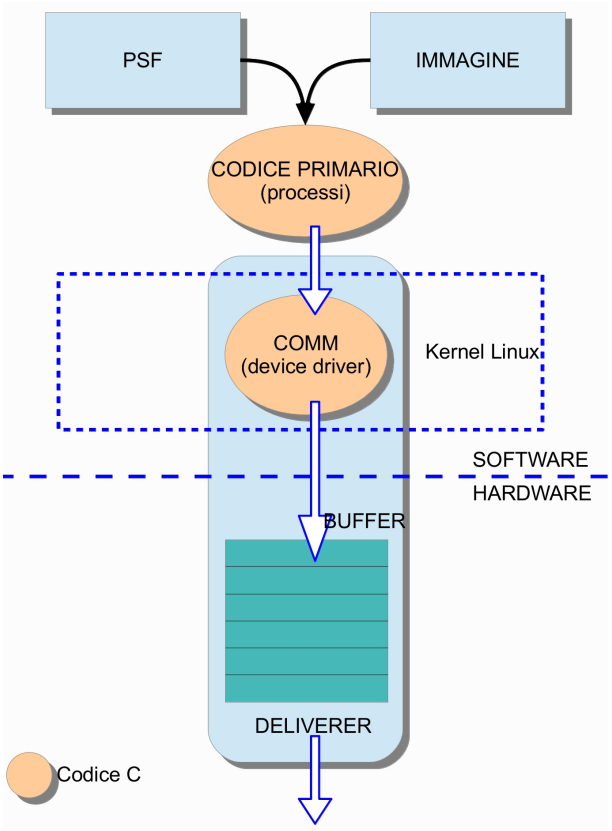

Figura 5.1: *Struttura del firmware.*

esempio le periferiche da abilitare, lo spazio di indirizzamento del processore, i blocchi IP da aggiungere e molte altre.

La configurazione del sistema Zynq adottata viene mostrata in figura 5.2. Il sistema programmabile PS viene interfacciato con la logica programmabile attraverso lo standard AXI. Tutte le periferiche connesse attraverso uno stesso blocco di interconnessione AXI-interconnect sono distinguibili attraverso un set di indirizzi specifico e assegnato in fase di configurazione. Ad uno stesso blocco di interconnessione vengono collegati un dispositivo detto master, il cui compito è quello di arbitrare la comunicazione sul bus, e uno o più dispositivi slave. Il processore ARM è un dispositivo master che può essere collegato ad altre periferiche solo attraverso l'AXI4-Lite che, come visto, non offre le migliori prestazioni in termini di velocità di trasferimento. Questa circostanza ha imposto la soluzione architetturale basata sul CDMA

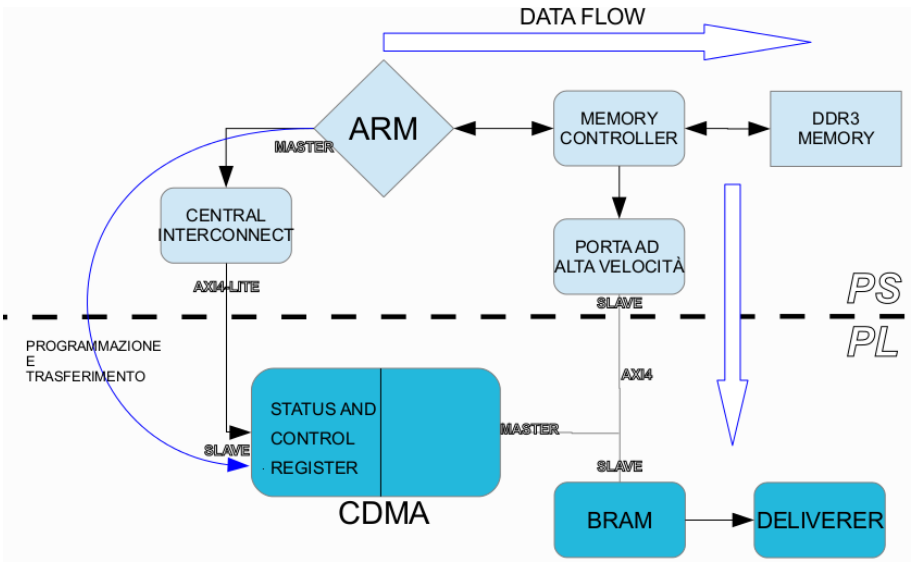

Figura 5.2: *Configurazione dello Zynq.*

di figura 5.2. poiché il flusso di dati prodotto dall'emulatore di posizione è superiore alla massima velocità teorica supportata dalla modalità Lite.

Il CDMA viene connesso a due blocchi di interconnessione distinti: sul primo, basato su AXI4-Lite, esso riveste il ruolo di dispositivo slave mentre sull'altro, basato su AXI4, di master. Il processore programma il trasferimento del CDMA effettuando la scrittura in opportuni registri di configurazione dello stesso, accessibili tramite risoluzione dell'indirizzo associato. Vista la soluzione architetturale, il trasferimento può coinvolgere solo periferiche collegate al CDMA lato master, ovvero attraverso la performante interfaccia AXI4. Questa rappresenta il punto di collegamento tra lo spazio PS e lo spazio PL siccome da un lato abbiamo la connessione con la memoria primaria del sistema, attraverso le porte ad alta velocità, e dall'altro con una memoria BRAM. In linea di principio possiamo immaginare la seguente dinamica di funzionamento: il processore sul quale è in esecuzione il codice relativo al position emulator produce un flusso di dati che viene temporaneamente salvato in un buffer di memoria. Periodicamente questo viene svuotato riversandone il contenuto, attraverso il CDMA, nella memoria BRAM. Questa operazione di bufferizzazione è necessaria per rendere efficiente il meccanismo di trasferimento. Il tempo necessario per programmare un trasferimento

risulta molto oneroso, in termini di cicli di clock del processore, utilizzando piccoli buffer. Consideriamo, per esempio, di poter disporre di un buffer molto piccolo e di uno molto grande; ad esempio in proporzione reciproca di $1 : N = 1\ dato : 10000\ dati$. Trasferendo una quantità di dati che riempie l'intero buffer grande sono necessari 10000 accessi al buffer piccolo ovvero 10000 programmazioni del trasferimento da parte del processore contro una sola per quello grande. Consideriamo che per la singola programmazione sia necessario un tempo t^{prog} e che per trasferire un singolo dato sia necessario t^{data}, per entrambi i buffer il tempo totale necessario a finalizzare l'operazione di trasferimento è semplicemente espresso dalla relazione $T = Nt^{data} + T^{prog}$ ove $T^{prog} = t^{prog}$ per la memoria grande e $T^{prog} = Nt^{prog}$per quella piccola. Appare evidente che se $t^{prog} \gg t^{data}$, come sempre accade, T è dominato dal tempo di programmazione per la memoria piccola; avere un buffer grande è sicuramente auspicabile.

D'altro canto l' FPGA integrata nello Zynq ha risorse di memoria decisamente limitate. Il compromesso che si è adottato è quello di allocare da 128Kword a 256Kword di BRAM al fine di preservare la restante memoria per l'implementazione dell'architettura responsabile del trasferimento dati tra Zynq ed FPGA (deliverer).

La memoria BRAM non è semplicemente collegata all'interconnessione AXI. Grazie alla disponibilità di due porte è possibile utilizzarla come memoria FIFO (First In First Out) collegante l'AXI al blocco deliverer. La porta della BRAM collegata all'AXI è la porta in scrittura della FIFO mentre la porta collegata al deliverer è quella in lettura.

Il software che si è implementato gestisce il sistema così configurato attraverso l'ausilio di un sistema operativo. Ciò è stato necessario per via della complessità associata alla gestione del parallelismo di esecuzione. Almeno tre, infatti, sono le operazioni fondamentali da compiere durante l'elaborazione: comunicazione con il PC host per interrompere, sospendere o riavviare l'emulazione, l'emulazione, ovvero la generazione dei dati, e la trasmissione dei dati stessi. Un sistema operativo illude il programmatore di poter eseguire simultaneamente, senza particolare sforzo, un numero anche abbastanza grande di operazioni.

Si è compreso fin da subito che nel panorama dei sistemi operativi compatibili con la scheda Zedboard la flessibilità richiesta dell'applicazione poteva essere soddisfatta soltanto da software opensource. Ecco che si è deciso di adattare il sistema oprativo Ubuntu, derivante dalla solida distribuzione Debian, ormai consolidato ed ampiamente diffuso. In particolare una versione standard compilata per l'architettura in questione è stata alleggerita da ogni componente non strettamente necessario quale ad esempio il server grafico X. Si può ben dire che il sistema operativo risultante sia costituito dal Kernel Linux vero e proprio e da pochi altri programmi accessori (es. busybox, opensshd, ...).

5.2 Kernel Linux

Il sistema operativo Linux è un sistema operativo "multi processo" e "multi utente". Ciò significa che la sua struttura prevede che siano operativi più processi ("programmi") contemporaneamente ed anche che più utilizzatori possano usare in modo condiviso lo stesso sistema; in questo è analogo ai più noti sistemi operativi della famiglia Windows.

L'architettura di Linux è strettamente simile a quella di Unix: un kernel molto "piccolo" che contiene solo le funzioni fondamentali per la gestione delle risorse del computer (memoria, dischi, rete e altre periferiche) ed una larga collezione di programmi applicativi che l'utente sfrutta per operare sul sistema. Anche all'interno del kernel la struttura delle varie componenti (processi, sistema di gestione della memoria, file system, etc.) è direttamente derivata da quella di Unix, tanto che si può affermare a buon titolo che Linux è una delle varie implementazioni del sistema operativo Unix.

Linux è nato nel 1991 come progetto di tipo amatoriale sviluppato da Linus Torvald, allora studente universitario ad Helsinki, per realizzare un "kernel" simile a quello del sistema operativo Unix che funzionasse su CPU della famiglia Intel. Fin dall'inizio Torvalds decise di mettere a disposizione di chiunque volesse usarlo il software da lui scritto e per fare ciò utilizzò la rete Internet che stava in quegli anni iniziando la sua grande espansione. Molte persone hanno poi iniziato a collaborare al progetto Linux creando

così il più grande gruppo di sviluppo di software mai esistito. Nello stesso periodo la "Free Software Foundation" stava sviluppando il progetto "GNU": la riscrittura da zero di tutti i programmi applicativi che costituiscono il sistema operativo Unix in modo da poter disporre di un sistema operativo "Unix like" completo e libero da licenze ed altri tipi di protezione. L'unione delle due cose ha portato al sistema operativo Linux come lo conosciamo e che dovrebbe a buon titolo essere chiamato GNU/Linux.

Iniziato come si è detto come progetto amatoriale, il sistema Linux ha avuto uno sviluppo estremamente interessante: la disponibilità dei sorgenti in forma "libera" ha stimolato moltissimi programmatori a produrre software in ambiente Linux tanto che praticamente tutte le applicazioni di tipo innovativo in campo informatico vengono sviluppate in ambiente Linux (o comunque Unix) prima di essere "rilasciate" sul mercato. Tra le caratteristiche salienti di Linux ricordiamo:

- Multitasking: possibilità di eseguire più programmi contemporaneamente.
- Funzionamento multiutente: uso contemporaneo della macchina da parte di più utenti.
- Multipiattaforma: Linux è praticamente compatibile con tutti i comuni hardware, e può girare anche su processori non INTEL (Digital Alpha, Sun Sparc, Mips, Motorola, PowerPC, ARM).
- funzioni di protezione della memoria tra processi.
- gestione della memoria virtuale attraverso la paginazione.
- un pool di memoria unificato per i programmi e la memoria cache
- librerie statiche e dinamiche
- file system proprio in cui i file possono avere nomi composti da non più di 255 caratteri e dimensioni fino a 2 Gbyte, con file system fino a 4 Tbyte.

- Possibilità di accedere a file system diversi: MS-DOS, VFAT (Windows 9x con nomi lunghi), HPFS (OS/2 2.x), NTFS (Windows NT - sola lettura), HFS (Apple Mac), FFS (Amiga), Minix, NFS (file system di rete), SMB (file system delle reti Windows) e così via.

- supporto per un'ampia gamma di protocolli di rete (TCP/IP, SLIP, PPP, etc...)

Se il kernel è il nucleo, la shell è il guscio. Per shell si intende l'interfaccia (testuale) tramite la quale l'utente può operare sul sistema. La shell è un programma che gestisce la comunicazione fra utente e sistema operativo interpretando ed eseguendo i comandi dell'utente. Può avere diversi utilizzi:

- Uso interattivo: il sistema attende i comandi digitati dall'utente;

- Configurazione della propria sessione: si possono definire variabili e parametri che vengono utilizzati in ogni interazione dell'utente con la macchina;

- Programmazione: utilizzando comandi di sistema e funzionalità della shell è possibile realizzare piccoli programmi (script shell) in grado di automatizzare operazioni e reagire ad eventi. La shell è qualcosa di simile all'interprete dei comandi del DOS, ma è molto più potente e versatile. In Linux sono disponibili diverse shell, ma la più diffusa è la bash.

In Linux i file rivestono un ruolo fondamentale. Un file non è solo un documento di testo o un programma eseguibile ma persino una periferica. La gestione dei file avviene attraverso specifici driver. Nel caso in cui il file sia associato ad una periferica allora un programma che fa accesso a quel file ricorre indirettamente al driver di dispositivo (device driver) associato. Per quanto riguarda la gestione dei file, alcune differenze fondamentali di Linux rispetto al DOS (e a Windows), sono le seguenti:

- I nomi dei file possono raggiungere la lunghezza massima di 255 caratteri

- Il carattere di separazione delle directory è il carattere "/" anziché il carattere "\"
- Linux distingue tra lettere minuscole e maiuscole (un file che ad esempio si chiama pippo.txt è diverso da un file denominato Pippo.txt nella stessa directory)
- I file (e directory) i cui nomi cominciano con un punto sono considerati file nascosti (generalmente sono utilizzati per contenere i parametri di configurazione del software personalizzati per ciascun utente)
- L'estensione dei file è spesso utile per determinare il tipo del file, ma non ha significato vincolante per il sistema operativo: i file eseguibili, generalmente, non hanno estensione
- I file e le directory hanno un proprietario (l'utente che ha creato il file) e un gruppo di appartenenza
- Come in Unix, i file e le directory hanno dei permessi: permessi di lettura, di scrittura, e di esecuzione. Tali permessi sono suddivisi in tre tipi (per ciascun file): i permessi concessi al proprietario del file, i permessi concessi al gruppo, i permessi concessi a tutti gli altri utenti
- Solo il proprietario del file (e l'amministratore di sistema) può modificare i permessi di un file
- Per poter lanciare un programma, occorre avere permessi di esecuzione sul file che contiene il programma
- Per poter accedere al contenuto di una directory, occorre avere permessi di lettura ed esecuzione sulla directory.

5.2.1 Gestione delle risorse

Essendo Linux un SO multitasking spesso accade che più processi in esecuzione cerchino di compiere operazioni in modo concorrente. Ogni processo richiede risorse di sistema quali ad esempio potenza di calcolo, memoria o

connettività di rete e si aspetta che queste siano ad esso riservate. Il kernel deve soddisfare le richieste dei processi e a tal fine è in grado di compiere le seguenti operazioni di base:

- Creare e distruggere processi e gestire le loro connessioni col mondo esterno (attraverso i device driver)
- Far comunicare due o più processi fra di loro (ad esempio attraverso pipe interprocesso, segnali o semafori).
- Illudere ogni processo di essere l'unico in esecuzione ed avere a disposizione l'intero sistema (per es CPU e memoria).

La memoria del sistema è una delle risorse principali, e la politica utilizzata per gestirla rappresenta una delle maggiori criticità per le performance dell'intero sistema. Il kernel costruisce uno spazio di indirizzamento virtuale, che verrà utilizzato da tutti i processi, al di sopra delle risorse fisiche limitate. Le diverse componenti del kernel interagiscono con il sottosistema di gestione della memoria attraverso una serie di funzioni specializzate (per esempio le note funzioni C di libreria standard malloc()/free()).

Linux è fortemente basato sul concetto di filesystem; quasi ogni cosa in Linux può essere trattata come un file. Il kernel costruisce un filesystem strutturato al di sopra dell'hardware non strutturato e la conseguente astrazione attraverso i file viene utilizzata nell'intero sistema. Persino le periferiche vengono individualmente viste come dei file o meglio *nodi* all'interno del filesystem. Il file in questo caso rappresenta il canale di comunicazione dei programmi con il gestore di dispositivo associato denominato *device driver.*

La connettività deve essere gestita dal sistema operativo in quanto la maggior parte delle operazioni di rete non sono specifiche per un determinato processo. In una rete di comunicazione tipicamente i dati vengono trasportati in pacchetti che possono giungere al sistema in qualsiasi istante. I pacchetti devono essere raccolti, identificati e instradati prima che un certo processo ne sfrutti il contenuto. E' il sistema operativo che si fa carico della trasmissione dei pacchetti di dati tra i programmi e le interfacce di rete assicurando tra l'altro l'esecuzione dei processi secondo le loro attività di rete.

5.2.2 Memoria reale e virtuale

Ogni volta che un microprocessore deve accedere alla memoria, per eseguire una istruzione di programma o per scambiare dati tra i registri e la memoria esterna, l'unità di calcolo richiede la lettura o scrittura del dato ad un indirizzo a 32 bit; tale indirizzo è cioè un numero compreso tra 0 e 4GB.

Perché il trasferimento del dato abbia effettivamente luogo, verranno inviati dei segnali elettrici su un bus esterno, in modo che qualche altro componente (memoria RAM o dispositivo periferico) riconosca la richiesta e risponda in modo appropriato. Sul bus esterno vengono normalmente posti oltre al dato anche 32 bit di indirizzo, utilizzati per identificare sia il componente esterno sia la parola di memoria cui accedere al suo interno.

Questi due indirizzi, nonostante siano entrambi numeri binari di 32 bit, non sono lo stesso numero. Il primo si chiama indirizzo virtuale mentre il secondo si chiama indirizzo fisico (o reale).

Durante il normale funzionamento del sistema, tutti gli indirizzi di memoria usati dai programmi sono indirizzi virtuali, senza alcuna eccezione, sia quando si lavora in spazio utente (nel caso ad esempio di un generico programma applicativo) sia quando si lavora in spazio kernel (es. device driver). Tutti i componenti esterni all'unità di calcolo, invece, rispondono agli indirizzi fisici sul bus, indipendentemente da quale indirizzo virtuale sia stato usato per generarli.

La memoria virtuale è un modo per costruire i 4GB di indirizzamento di un programma a proprio piacimento, in base alle esigenze di programmazione senza dipendere troppo dalla struttura fisica della macchina su cui si lavora.

Lo spazio virtuale, i 4GB indirizzabili dal processore, è separato in due parti: gli indirizzi più bassi, normalmente fino a 3GB, sono a disposizione del processo, mentre gli indirizzi più alti sono "privilegiati": l'accesso è consentito al solo nucleo del sistema operativo. Poiché il kernel è il tramite della comunicazione tra i processi e dell'accesso al filesystem, ha bisogno di un'area di memoria condivisa che sia sempre accessibile, sia eseguendo le chiamate di sistema per conto dei processi sia durante la gestione delle interruzioni. L'ultimo gigabyte di spazio virtuale, perciò, contiene il codice del kernel, tutte le sue strutture dati e la memoria delle periferiche cui devono accedere i

driver di sistema. Per semplicità, la prima parte di questo GB è direttamente mappata sulla memoria RAM del sistema.

La costruzione dello spazio virtuale per gli indirizzi non privilegiati è invece a totale discrezione del processo. Associando gli indirizzi virtuali agli indirizzi reali si costruisce la *mappa di memoria* per un certo processo. L'associazione non è vincolante; un processo può modificare la sua mappa di memoria purché ciò non interferisca con il corretto funzionamento del kernel. L'operazione di riassegnamento è di fondamentale importanza quando si ha la necessità di accedere ad un particolare indirizzo reale. Il meccanismo comunemente usato dai processi per modificare la propria mappa virtuale è la chiamata di sistema mmap, con la quale si richiede al sistema operativo di vedere nel proprio spazio di memoria il contenuto di un file o di un dispositivo, oppure ottenere indirizzi di memoria anonima, cioè RAM, non associata ad un file.

5.2.3 I processi

Un programma eseguibile, generato ad esempio da un programma in codice C, è rigorosamente sequenziale, nel senso che viene eseguito una istruzione alla volta e, dopo l'esecuzione di un'istruzione, è univocamente determinata quella successiva da eseguire. L'esecuzione di N programmi da parte di un calcolatore dovrebbe anch'essa essere rigorosamente sequenziale, in quanto, dopo avere eseguito la prima istruzione di un programma dovrebbero essere eseguite tutte le successive fino alla fine del programma prima di poter iniziare l'esecuzione della prima istruzione del programma successivo. Questo modello sequenziale è molto comodo per il programmatore, perché nella scrittura del programma egli sa che non ci saranno "interferenze" da parte di altri programmi, dato che questi verranno eseguiti, dallo stesso esecutore, prima o dopo l'esecuzione del programma considerato. Tuttavia, il modello di esecuzione sequenziale non è adeguato alle esigenze della maggior parte dei sistemi di calcolo; questa inadeguatezza è facilmente verificabile pensando ai seguenti esempi:

- **Server WEB**: un server WEB deve poter rispondere a molti uten-

ti contemporaneamente; non sarebbe accettabile che un utente, per collegarsi, dovesse attendere la disconnessione di tutti gli altri utenti.

- **Calcolatore multiutente**: i calcolatori potenti vengono utilizzati da molti utenti contemporaneamente; in particolare, i calcolatori centrali dei Sistemi Informativi (delle banche, delle aziende, ecc...) devono rispondere contemporaneamente alle richieste di moltissimi utilizzatori contemporanei

- **Applicazioni multiple aperte da un utente**: quando un utente di un normale PC tiene aperte più applicazioni contemporaneamente esistono diversi programmi che sono in uno stato di esecuzione già iniziato e non ancora terminato.

In base ai precedenti esempi risulta necessario un modello più sofisticato del sistema; si osservi che tale modello deve garantire due obiettivi tendenzialmente contrastanti:

- fornire parallelismo, cioè permettere che l'esecuzione di un programma possa avvenire senza attendere che tutti gli altri programmi in esecuzione siano già terminati;

- garantire che ogni singolo programma in esecuzione sia eseguito come nel modello sequenziale.

Per ottenere un comportamento del sistema che soddisfi gli obiettivi indicati sopra la soluzione più comune è quella rappresentata in figura 5.3; notiamo la presenza di tanti esecutori quanti sono i programmi da essere eseguiti. Nel contesto del sistema operativo Linux gli esecutori creati dinamicamente per eseguire diversi programmi sono detti processi. I processi devono essere considerati degli esecutori completi, e quindi la struttura rappresentata in figura 5.3 risponde in maniera evidente ad ambedue i requisiti sopra illustrati: al primo, perché i diversi processi eseguono programmi diversi in parallelo, cioè senza che uno debba attendere la terminazione degli altri, e al secondo requisito perché, essendo i diversi processi degli esecutori indipendenti tra loro, non c'è interferenza tra i diversi programmi (come se fossero eseguiti su diversi calcolatori).

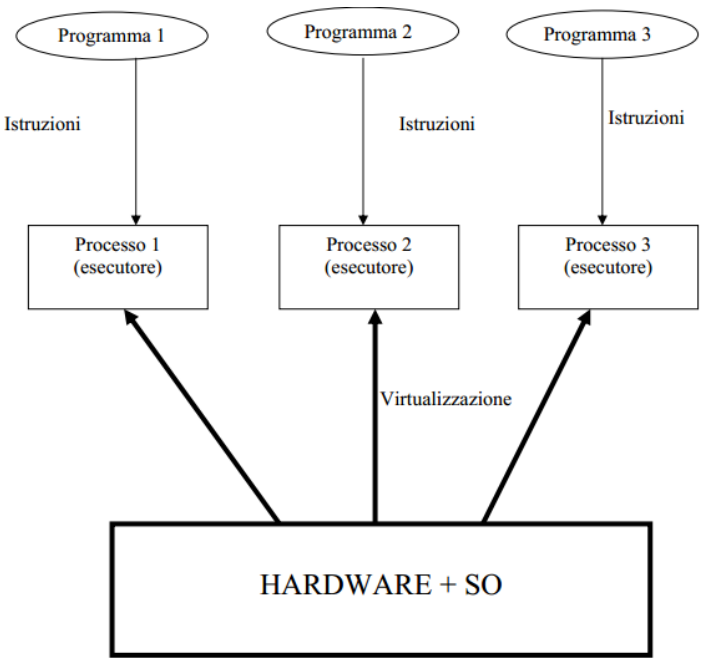

Figura 5.3: *I processi su Linux.*

I processi possono essere considerati come dei calcolatori o macchine virtuali, nel senso che sono calcolatori realizzati dal software (sistema operativo) e non esistono in quanto hardware, anche se ovviamente il sistema operativo ha a sua volta bisogno di essere eseguito da un calcolatore reale. Ogni processo deve possedere ad ogni istante un unico programma in esecuzione; pertanto, la comunicazione tra due processi coincide con la comunicazione tra i due corrispondenti programmi in esecuzione. Una delle principali risorse utilizzate dai processi è la memoria che viene suddivisa in tre parti:

- il segmento codice (text segment): contiene il codice eseguibile del programma lanciato in esecuzione sul processo
- il segmento dati (user data segment): contiene tutti i dati del programma, sia i dati statici, sempre presenti, sia i dati dinamici, che a loro volta si dividono in dati allocati automaticamente in una zona di memoria detta pila (stack), che sono i record di attivazione delle funzioni (variabili locali, indirizzo di ritorno e parametri delle funzioni), e in da-

ti allocati esplicitamente dal programma tramite la funzione "malloc( )" in una zona di memoria detta heap.

- il segmento di sistema (system data segment): questo segmento contiene dati inerenti il processo stesso (ad esempio, la "tabella dei files aperti") che tuttavia non sono gestiti esplicitamente dal programma in esecuzione ma dal sistema operativo per suo conto. Si noti che esistono strutture dati necessarie alla gestione del singolo processo che non sono allocate in questa area, ma in strutture dati interne del sistema operativo (ad esempio, il descrittore del processo nella tabella dei processi)

Ogni volta che un programma viene lanciato in esecuzione si ha la creazione del relativo processo. Il programma lanciato, detto padre, può richiedere al sistema operativo la creazione di uno o più processi figli tramite l'utilizzo di opportune funzioni di sistema (fork). I processi figli assomigliano al processo padre eccetto che per un descrittore detto pid attraverso il quale è possibile differenziare l'esecuzione dei processi. E' importante sottolineare che siccome i processi figli sono indipendenti dal padre per ognuno di essi è riservata memoria e la loro esecuzione può continuare ben oltre la terminazione del padre.

Spesso accade che i processi abbiano l'esigenza di scambiarsi informazioni. Questa operazione può essere compiuta attraverso diverse modalità che possiamo raggruppare in due categorie; *lo scambio di messaggio* e l'utilizzo di *memoria condivisa.* Sebbene quest'ultima modalità sia la più intuitiva essa è la meno auspicabile in quanto viola il principio di indipendenza tra i processi che sta alla base dell'esecuzione multiprocesso.

Lo *scambio di messaggi* è sicuramente la tecnica più elegante per sincronizzare e collegare più processi. Tale tecnica si basa sull'utilizzo di strutture dati messe a disposizione dal sistema operativo. In Linux sono disponibili diverse tipologie di messaggi ma le principali sono sicuramente i *segnali,* le *pipe* e i *semafori.* Un segnale, non trasportando l'informazione relativa al mittente, è il metodo più semplice per inviare un messaggio; esso consiste di un codice a cui è associato un comando.

Una pipe come suggerisce il termine è un canale di comunicazione tra due processi. Una volta creata permette l'invio seriale di dati tra un processo sorgente ed uno destinatario secondo lo schema classico di una FIFO. Ogni trasferimento coinvolge l'intervento del sistema operativo che ha il compito di ricevere il messaggio da un processo e consegnarlo all'altro. Per questo motivo la comunicazione è sicura e non si hanno interferenze in memoria tra i processi.

La tecnica dei semafori, diversamente dalla precedente, viene utilizzata quando non si ha la necessità di comunicare dati veri e propri ma piuttosto un'informazione relativa allo stato di un processo. Un tipico esempio di uso dei semafori è la sincronizzazione dell'esecuzione. N processi che realizzano insieme una certa elaborazione possono essere eseguiti a velocità molto diverse sul sistema così che questi giungano ad istanti diverse ad una sezione critica del codice. Possiamo immaginare che in questa sezione critica si raccolgano tutti i dati elaborati dagli N processi. L'istante in cui bisogna raccogliere i dati corrisponde al momento in cui il processo più lento giunge all'inizio della sezione critica. Poiché l'istante non è a priori noto ogni processo deve comunicarlo a tutti gli altri; questo si realizza attraverso un semaforo. In buona sostanza un certo semaforo non è nient'altro che un dato di stato (o flag) visibile da tutti i processi che ne fanno uso.

5.2.4 Device driver in Linux

Uno dei tanti vantaggi offerti dai sistemi operativi open source, cui Linux appartiene, è che il loro codice è completamente visionabile, comprensibile e modificabile; Linux ha permesso la democratizzazione dei sistemi operativi. Sebbene il kernel Linux sia costituito da un'enorme mole di codice relativamente complesso spesso si ha la necessità di sviluppare parte di esso senza la necessità di avere una sua visione onnisciente. Questo si verifica nel caso dello sviluppo dei device driver. Questi programmi hanno un ruolo fondamentale all'interno del sistema operativo: forniscono il supporto software per un particolare dispositivo hardware connesso al sistema. Essi mascherano i meccanismi di funzionamento del dispositivo fornendo un'interfaccia standard di

comunicazione con esso. L'utente, infatti, può utilizzare un certo dispositivo, ignorando totalmente come esso funzioni, affidandosi semplicemente ad una serie di *primitive di comunicazione* definite una volta per tutte.

Un device driver può essere "inserito" e "disinserito" all'interno del kernel in qualsiasi momento, offrendo la possibilità di selezionare la combinazione ottimale di driver in funzione dell'hardware di sistema. Questa modularità rende la scrittura del codice dei device driver abbastanza semplice visto che, almeno in prima istanza, solo due sono le condizioni necessarie: è necessario conoscere bene come il dispositivo funziona e come il sistema operativo vuole che esso sia visto da parte dell'utente.

Il ruolo di un device driver è quello di fornire un *meccanismo di accesso* ad una certa periferica e non una *politica di gestione* della stessa. Nello scrivere un device driver bisogna sempre avere in mente che ogni utente può voler gestire il dispositivo in base alle sue esigenze. Per fissare le idee si può pensare alla realizzazione di un driver per un disco di memoria. Almeno due sono le operazioni fondamentali che un driver deve essere in grado di compiere; la lettura e la scrittura. La lettura e la scrittura sono tipici esempi di meccanismi di accesso alla periferica. Una politica di gestione invece è la visione virtuale che si ha del disco attraverso il filesystem.

Una delle peculiarità di Linux è la possibilità di estendere runtime le capacità del kernel. Ogni segmento di codice che può essere aggiunto al kernel runtime viene chiamato modulo. Linux supporta un gran numero di tipologie differenti di moduli come per esempio quella dei device driver. Ogni modulo non è un vero e proprio programma eseguibile. Esso è un codice oggetto che viene unito al codice del kernel dinamicamente attraverso il linking. Tale operazione è realizzata attraverso il programma insmod mentre quella complementare di de-linking attraverso rmmod.

In Linux un qualsiasi dispositivo può essere considerato appartenente ad una delle tre classi di dispositivi riconosciute dal sistema. Alle tre classi di dispositivi corrispondono altrettante classi di device driver. Ogni modulo tipicamente implementa una sola di queste tipologie ma è comunque possibile realizzare un enorme modulo al cui interno implementare driver differenti. Sebbene possibile, tale tecnica non è consigliabile in quanto limita fortemente

la modularità delle risorse. Le tre classi di dispositivi sono le seguenti:

- **Dispositivi a char.** Un dispositivo a caratteri può essere immaginato come un generico file. L'accesso viene effettuato attraverso la lettura o la scrittura di singoli byte. Al device driver a char spetta il compito di portare a termine queste operazioni attraverso le due funzioni base read e write. Le funzioni aggiuntive open e close insieme alle precedenti due costituiscono la base della comunicazione dei processi con il device driver. L'accesso ai dispositivi a char deve passare necessariamente attraverso l'apertura, con la funzione open, del nodo di dispositivo all'interno del filesystem. Completati i trasferimenti è necessario comunicare al sistema operativo, o meglio al device driver, il termine delle operazioni attraverso la funzione close. La differenza fondamentale rispetto ad un generico file è che nella maggior parte dei dispositivi a caratteri l'accesso ai byte è sequenziale e quindi non è possibile muoversi avanti ed indietro all'interno di un'area di di dati.

- **Dispositivi a blocchi.** Come per i dispositivi a caratteri anche per quelli a blocchi l'accesso viene effettuato attraverso nodi nel filesystem. Un dispositivo a blocchi è sostanzialmente un dispositivo che può ospitare un filesystem. Nella maggior parte dei sistemi Unix un dispositivo a blocchi può gestire solo operazione di input-output che trasferiscono uno o più blocchi di dimensione 512 byte (o una potenza di due molto maggiore). Linux invece permette ad un processo la lettura o la scrittura di un numero arbitrario di byte per ogni trasferimento. Conseguentemente i dispositivi a blocchi e a caratteri differiscono solo nel modo in cui i dati vengono gestiti dal driver all'interno del kernel. Sebbene la differenza tra le due tipologie di driver sia sottile guardandola dal punto di vista del processo utilizzatore, i driver a blocchi si interfacciano in un modo completamente diverso con il kernel rispetto a quelli a char.

- **Interfacce di rete.** Qualsiasi transazione di rete viene effettuata attraverso un'interfaccia, cioè un dispositivo capace di scambiare dati, con altri dispositivi ad esso equivalenti, all'interno di una infrastruttura

di rete. Di norma un'interfaccia è un dispositivo fisico ma essa può anche essere simulata via software (interfaccia di loopback). Il compito di un'interfaccia di rete è di trasmettere e ricevere pacchetti di dati, veicolati attraverso il sottosistema di rete del kernel, mappandoli sulle singole transazioni di rete. Questi dispositivi difficilmente si mappano all'interno del filesystem come visto per i precedenti due, tuttavia il kernel nomina ogni interfaccia attraverso un proprio identificativo (es. eth0).

5.3 Implementazione emulatore posizione

L'emulazione della posizione viene realizzata attraverso un codice scritto nel linguaggio di programmazione C che viene compilato per essere eseguito in modalità utente sul sistema operativo Ubuntu.

L'elaborazione ha inizio caricando in memoria il vettore cumulativo rappresentante l'immagine da emulare. Questo viene creato attraverso uno script Matlab a partire dall'immagine illustrata in figura 3.9 e viene memorizzato in un file binario. Questa operazione potrebbe essere implementata sullo Zynq caricando direttamente l'immagine ma si è preferito procedere in questo modo per mantenere una certa flessibilità nella conversione (ad esempio per riuscire a sintetizzare rapidamente una nuova immagine). Il file binario ottenuto viene salvato all'interno del filesystem di Linux in esecuzione sul dispositivo e passato in ingresso al programma di emulazione. Vista l'architettura a 32 bit dell'ARM è stato conveniente normalizzare il vettore cumulativo al valore massimo 2^{32}.

Oltre al file contenente il vettore cumulativo il programma riceve in ingresso un file contenente l'informazione relativa alle PSF. Diversamente da quanto visto nel capitolo 3 dove ad ogni detector emulato corrispondeva una specifica PSF, il file in questione contiene 256*256 nuove PSF che chiameremo audacemente PSF^*. Ogni PSF^* non è più associata ad un certo detector ma ad un certo pixel della matrice immagine; ve ne sono infatti 256*256 ovvero il numero totale di punti di incidenza gamma che si possono emulare. La PSF^* è una matrice di 8*8 parole a 16 bit e ci indica la carica competente

ad ogni detector per ciascun pixel di incidenza. L'approccio appena descritto viene detto a PSF interlacciate poiché derivante dalla concatenazione dell'informazione contenuta nelle varie PSF.

Il procedimento di costruzione di una generica PSF^*è abbastanza semplice. Ipotizziamo, per esempio, di dover trovare le cariche dei 64 detector nel caso in cui sia stata estratta la posizione $(5, 128)$ ovvero la $PSF^*_{5,128}$. Accedendo alle 64 PSF attraverso la posizione $(5, 128)$ si raccolgono i 64 valori che costituiscono proprio la $PSF^*_{5,128}$. Sia il vettore cumulativo che l'insieme delle PSF^*vengono salvate in memoria allocando 2 buffer di dimensione opportuna (circa 8MB per PSF^*e poche centinaia di KB per il cumulativo).

Il processo di emulazione della posizione può essere scomposto in tre semplici fasi che si ripetono ciclicamente all'infinito:

- **RNG.** Al fine di massimizzare la velocità di esecuzione un generatore lineare congruenziale estrae un numero pseudo casuale di valore massimo $MAX = 2^{32}$ in modo tale che MAX coincida con il massimo valore presente nel vettore cumulativo.

- **Emulazione posizione.** Il numero estratto viene ricercato attraverso un algoritmo di ricerca binaria all'interno del vettore cumulativo con la procedura consolidata. Alla fine si ottiene l'indice del vettore che corrisponde alla posizione di incidenza gamma emulata.

- **Generazione carica.** Attraverso la posizione emulata si accede alla zona di memoria contenente le PSF^*generando le cariche dei 64 detector per questa incidenza.

L'implementazione è decisamente più complessa dei tre passaggi illustrati. Per poter generare un flusso di dati paragonabile con quello di comuni gamma camera si sono sfruttate le potenzialità offerte dal sistema operativo. A tal scopo vengono creati un processo padre ed un processo figlio che, vista la particolare architettura dual core del processore ARM, sono praticamente eseguiti in parallelo. Sebbene il flusso di elaborazione sia identico per entrambi i processi essi operano su dati differenti implementando in modo implicito un sistema di calcolo SIMD. La diversità dei dati origina dalla generazione di differenti numeri casuali.

Siccome un processo figlio eredita una copia della memoria del padre nel momento in cui viene creato, la generazione di una sequenza casuale per i due processi non è così semplice. Affinché un RNG fornisca sequenze di valori sempre diverse è necessario garantire semi di buona qualità. Questo si può ottenere ricorrendo all'orologio di sistema che per interrogazioni successive restituisce valori sempre diversi (l'ora continua a crescere).

La soluzione adottata prevede l'utilizzo di due sequenze pseudocasuali identiche, poiché originate dallo stesso seme (inizio dell'emulazione), sia per il padre che per il figlio. Naturalmente se ogni processo lavorasse sugli stessi numeri casuali lo sforzo di parallelizzare l'elaborazione sarebbe completamente inutile. Per questo motivo (vedi figura 5.4) un processo ha l'onere di elaborare tutte le estrazioni pari della sequenza e l'altro quelle dispari. Il risultato finale è che una certa sequenza pseudocasuale viene riprodotta in circa la metà del tempo necessario per un sistema single core.

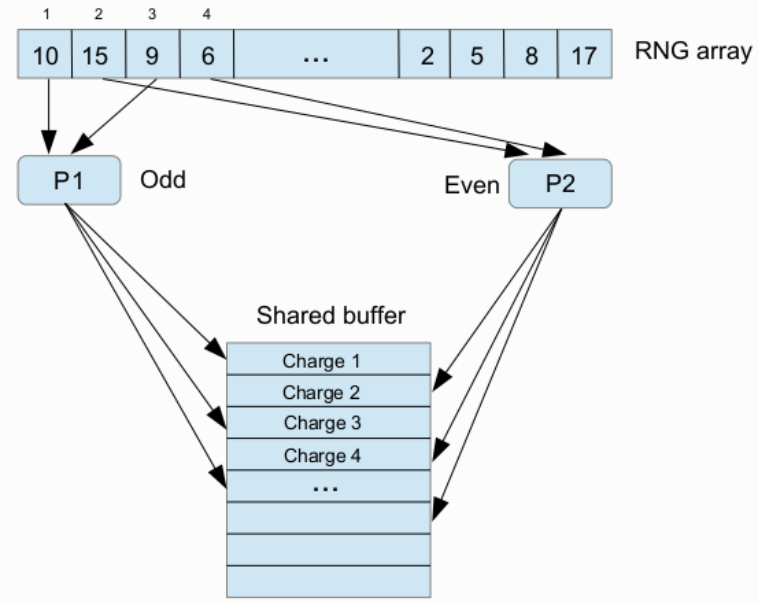

Figura 5.4: I *due processi P1 e P2 elaborano differenti estrazioni della stessa sequenza.*

Le matrice di carica (ad esempio charge1 in figura) che ogni processo crea per una certa estrazione non è altro che una copia della PSF^*associata alla posizione sorteggiata. Le matrici di carica ottenute vengono salvate in unico

buffer di memoria secondo l'ordine di estrazione; un processo genererà la prima, la terza, la quinta matrice carica e così via mentre l'altro la seconda, la quarta ecc. Dai due processi si crea così un unico stream di dati che potrà essere trasmesso al deliverer.

Tra le implementazioni possibili l'utilizzo di un'area di memoria condivisa tra i processi è l'unica praticabile. Infatti utilizzando un unico buffer privato nell'area di memoria del padre ed una pipe per l'invio delle matrici charge elaborate dal figlio verrebbe richiesto un tempo eccessivo per lo scambio dei dati rallentando pesantemente il rate di emulazione. La memoria condivisa rappresenta la soluzione vincente soprattutto perché associata ad un meccanismo ottimizzato di copia dei dati che verrà ora illustrato.

Nell'architettura di calcolatore utilizzata la memoria e la CPU si interfacciano con bus dati a 32 bit. Ogni operazione di trasferimento sul bus impiega sempre la stessa quantità di tempo per essere finalizzata, a prescindere dal numero di bit trasmessi, quindi risulta vantaggioso sfruttare, per ogni transazione, l'intera larghezza del bus dati trasmettendo parole a 32 bit. Attraverso la tecnica dell'interlacciamento è possibile ottimizzare la copia delle matrici di carica. Sapendo che ogni valore (carica) delle PSF^*è un numero a 16 bit e che valori appartenenti alla stessa $PSF^*_{x,y}$ fanno parte della stessa emulazione, è possibile unire a due a due le parole da 16 bit consecutive (partendo dalla prima contenuta in $PSF^*_{x,y}$) in parole da 32 bit in modo tale che ogni matrice di carica occupi una dimensione di 32 parole da 32 bit piuttosto che 64 parole da 16 bit ognuna. Il vantaggio è evidente: si migliora la velocità di emulazione di un fattore due. Facciamo notare che il miglioramento è anche più importante rispetto all'utilizzo delle PSF non interlacciate. Se in memoria fosse caricato un vettore contenente le PSF standard vista la loro costruzione, il processore dovrebbe calcolare per ogni valore di carica da trasferire il relativo indirizzo sorgente e quello di destinazione mentre in questo caso il calcolo è molto più veloce: siccome sia gli indirizzi sorgente che quelli destinazione sono consecutivi essi possono essere calcolati incrementando, di un'unità, di volta in volta, gli indirizzi di base.

L'implementazione del buffer condiviso richiede particolare attenzione. Per via dell'indirizzamento virtuale utilizzato dal sistema operativo ogni vol-

ta che un processo richiede una certa area di memoria libera, questa viene ricercata all'interno dello spazio di indirizzamento virtuale riservato per quel processo. Nel momento in cui un processo figlio viene creato questo eredita una copia dello spazio di indirizzamento virtuale del padre. Essendo una copia, la memoria risultante conterrà i dati calcolati dal padre fino al momento della creazione. Per questo motivo facendo in modo che il padre richieda al SO un buffer in memoria prima della generazione del figlio, tale porzione di memoria sarà la stessa nello spazio di indirizzamento virtuale dei due processi. Tuttavia, essendo i due processi indipendenti, ogni scrittura sul buffer di un processo non è visibile all'altro; è necessario condividere il buffer creato.

La memoria fisica come qualsiasi altra periferica viene gestita in Linux attraverso un opportuno device driver. Il SO con l'indirizzamento virtuale implementa una *politica di gestione* della memoria fisica attraverso i *meccanismi di accesso* alla hardware garantiti dal device driver. Una delle funzionalità avanzate offerte dal device driver della memoria in congiunzione con il kernel stesso è quella di modificare la mappa di memoria per un certo processo. Utilizzando la primitiva mmap il processo padre ed il processo figlio possono rimappare il proprio buffer privato sulla stessa porzione di memoria fisica RAM; quando i due processi scrivono un dato in una cella di memoria attraverso uno stesso indirizzo virtuale, la rimappatura fa in modo che quel dato sia presente nella stessa cella di memoria fisica per entrambi i processi.

Si fa notare che nell'implementazione realizzata non sorgono mai conflitti di accesso alla porzione di memoria condivisa, cosa che genererebbe una eccezione ed un conseguente aborto dell'esecuzione. In altre parole, non è possibile che entrambi i core vogliano scrivere contemporaneamente nella stessa cella di memoria in quanto il riempimento del buffer compete, per le matrici di carica dispari, ad un processo e, per quelle pari, all'altro.

La rimappatura ha pro e contro. Sicuramente questa rappresenta un modo veloce ed agevole di scambiare dati sia tra processi che tra processi e periferiche ,siccome anche le periferiche vengono viste come indirizzi nello spazio di indirizzamento reale del processore. D'altra parte però scavalcando la gestione delle risorse del sistema possono sorgere problematiche altrimenti mascherate dal SO. Scrivere in modo diretto in zone di memoria non è

un'operazione così indolore; se una certa area di memoria contenente dati di altri processi o addirittura variabili di stato del sistema operativo viene sovrascritta i processi, o addirittura l'intero sistema, possono cessare il loro funzionamento.

Lo stratagemma adottato nell'implementazione sullo Zynq è stato quello di imporre al SO, in fase di boot del sistema, attraverso opportune configurazioni, una dimensione della memoria RAM inferiore a quella effettiva. In questo modo Linux utilizzerà soltanto la prima porzione della memoria primaria lasciando completamente inutilizzata quella terminale. Quest'ultima porzione, quindi, verrà utilizzata come buffer condiviso tra i due processi di emulazione.Come precedentemente illustrato il buffer viene periodicamente svuotato copiandone il contenuto nella memoria BRAM. La copia ha inizio quando i processi in esecuzione inviano un segnale di sincronismo al deliverer.

In realtà un solo processo ha il compito di inviare al deliverer il segnale di sincronismo nel momento in cui il buffer risulta pieno. Poiché i due processi riempiono il buffer con velocità non necessariamente uguali è obbligatorio sincronizzare l'esecuzione in modo tale che il trasferimento sia inizializzato solo dopo la scrittura dell'ultimo valore nel buffer. Tale meccanismo di sincronizzazione viene realizzato attraverso l'utilizzo di un barriera. La barriera è un sistema di sincronizzazione basato sulla struttura dei semafori fornita dal SO Linux e funziona in questo modo. Si utilizzano i due tipi di messaggio *sono arrivato alla barriera* e *non sono arrivato alla barriera,* ogni processo memorizza in una variabile locale lo stato relativo all'altro processo ed il processo padre (P) è responsabile dell'invio del segnale al deliverer. Supponiamo che inizialmente lo stato di entrambi i processi sia *non sono arrivato alla barriera.* Il figlio ogni volta che giunge alla sua ultima scrittura si mette in attesa e comunica a P *sono arrivato alla barriera.* Se nel frattempo anche P è arrivato alla barriera questo inizializza il trasferimento altrimenti si mette in attesa del figlio. Dopo l'inizializzazione il padre sblocca il figlio attraverso il messaggio *non sono arrivato alla barriera* e l'esecuzione ricomincia per entrambi.

5.3.1 Deliverer

Il deliver è, a tutti gli effetti, una periferica collegata al sistema e per questo la sua gestione è affidata ad un device driver. Il device driver effettua il trasferimento dei dati che il processo padre periodicamente inizializza una volta riempito il buffer di memoria. Questo obbiettivo è raggiunto utilizzando la configurazione della PL dello Zynq vista a inizio capitolo. Il driver una volta ricevuto il comando di inizializzazione da parte del processo padre non fa altro che programmare un trasferimento DMA, settando opportuni registri nel CDMA, tra il buffer in memoria primaria e la BRAM. I dati presenti in BRAM verranno poi trasmessi con un'opportuna logica dedicata all'FPGA Kintex7.

Il deliverer impiegherà un certo tempo per svuotare il buffer in memoria durante il quale i due processi non possono scrivere nuovi valori per non corrompere i dati di simulazioni precedenti. Al fine di ottimizzare le performance del trasferimento il buffer è stato diviso in tre parti uguali. In questo modo i due processi una volta riempito un sotto-buffer possono passare subito a scrivere valori nel successivo. Il terzo sotto-buffer disponibile viene utilizzato per creare una coda di trasferimento nel caso in cui il deliverer impieghi troppo tempo e i processi abbiano già riempito il secondo buffer.

Secondo questo meccanismo il processo padre inizializza il trasferimento semplicemente inviando al device driver il numero del buffer in memoria che deve essere trasferito.

La modalità DMA impostata è quella a "pacchetto di dati" ovvero la prima illustrata nella sottosezione 4.3.2 per concentrare il tempo di accesso alla ram in modo tale da interferire il meno possibile con il sistema operativo. I dati di emulazione una volta presenti in BRAM vengono trasmessi lungo la catena di emulazione.

Capitolo 6

Hardware

L'hardware dell'emulatore comprende una scheda Zedboard che svolge il ruolo di master del sistema e quattro schede slave che sono state sviluppate ad hoc. In questo capitolo si mostrerà l'architettura hardware delle schede slave. Poiché le quattro schede sono identiche ci si concentrerà sull'analisi del singolo prototipo realizzato.

La scheda (vedi figure 6.1 6.2) ospita solo una parte dei dispositivi che producono l'elaborazione globale in accordo con lo schema architetturale mostrato nel capitolo 4 fig. .

L'FPGA kintex7 rappresenta il cuore dell'elaborazione competente alla scheda. Oltre ad essa sono presenti 8 DAC e 16 stadi di condizionamento analogico.

L'FPGA e gli stadi di condizionamento definiscono due domini, uno digitale ed uno analogico, che sono nettamente distinti. Sulla scheda infatti si possono identificare le due aree indipendenti che vengono connesse solo in corrispondenza degli stadi di conversione D/A. Nella zona digitale trovano posto oltre all'FPGA diversi dispositivi necessari per il funzionamento del sistema. Una memoria flash, posizionata in prossimità dell'FPGA, viene utilizzata per programmare all'avvio il bitstream della Kintex7. La programmazione può essere effettuata anche attraverso un'interfaccia JTAG, particolarmente utile in fase di debug. Il clock di sistema a 125MHz è unico e viene distribuito all'FPGA ed agli 8 DAC attraverso una rete di piste differenziali. Il sistema è dotato di un'interfaccia USB accessoria che permette l'utilizzo della scheda

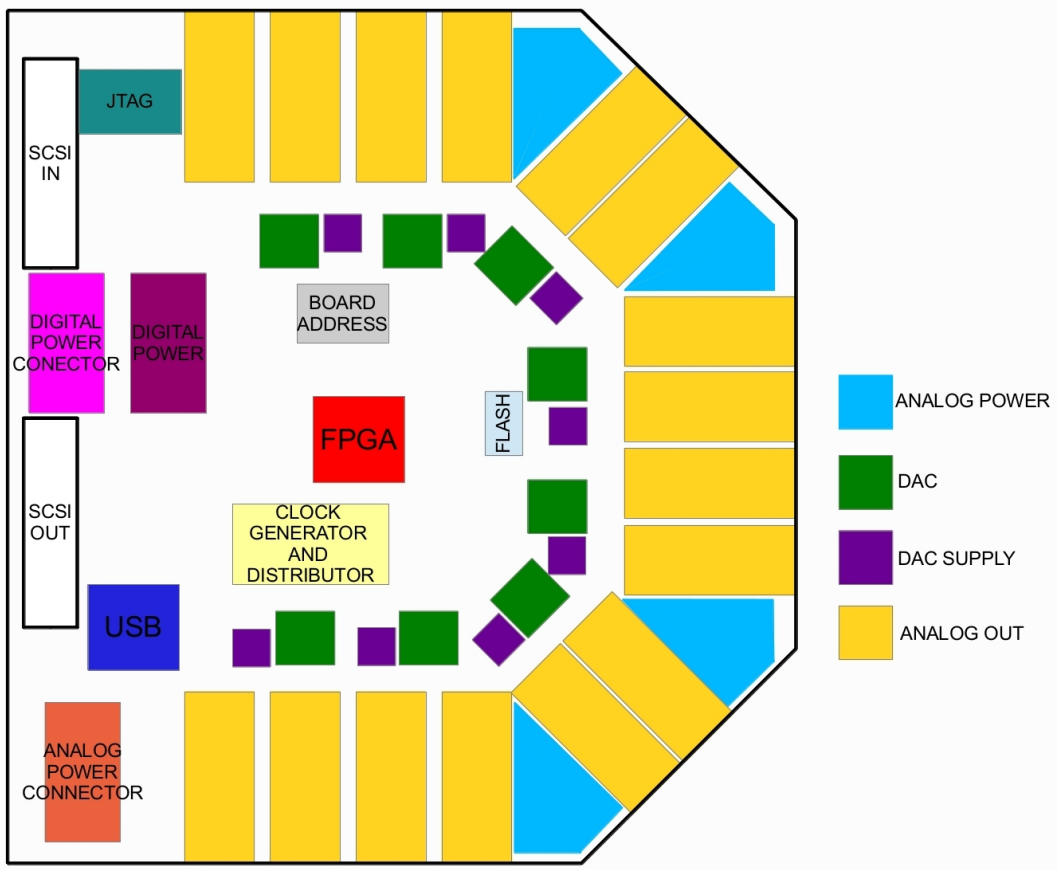

Figura 6.1: *Schema della scheda prototipale realizzata.*

in modalità general purpose.

Particolare importanza ha, nell'utilizzo dell'area della scheda, il routing delle piste che costituiscono il bus di interfacciamento della Kintex7 con l'FPGA master cioè lo Zynq. Come possiamo vedere dal layout 6.2 sono presenti sulla sinistra due connettori SCSI e ad ognuno è associato un bus. I bus sono di tipo parallelo sincrono con un clock secondario a 64Mhz fornito dallo Zynq ed il trasporto dei dati è di tipo differenziale.

Nel posizionamento dei dispositivi è stato scelto di inserire sul livello top della scheda tutti quelli relativi all'elaborazione digitale del segnale, mentre sul bottom vengono collocati i 64 stadi di condizionamento analogico al fine di separare attraverso vari piani di massa ed alimentazione i due domini presenti. L'interfacciamento dei DAC con l'FPGA avviene attraverso 8 bus

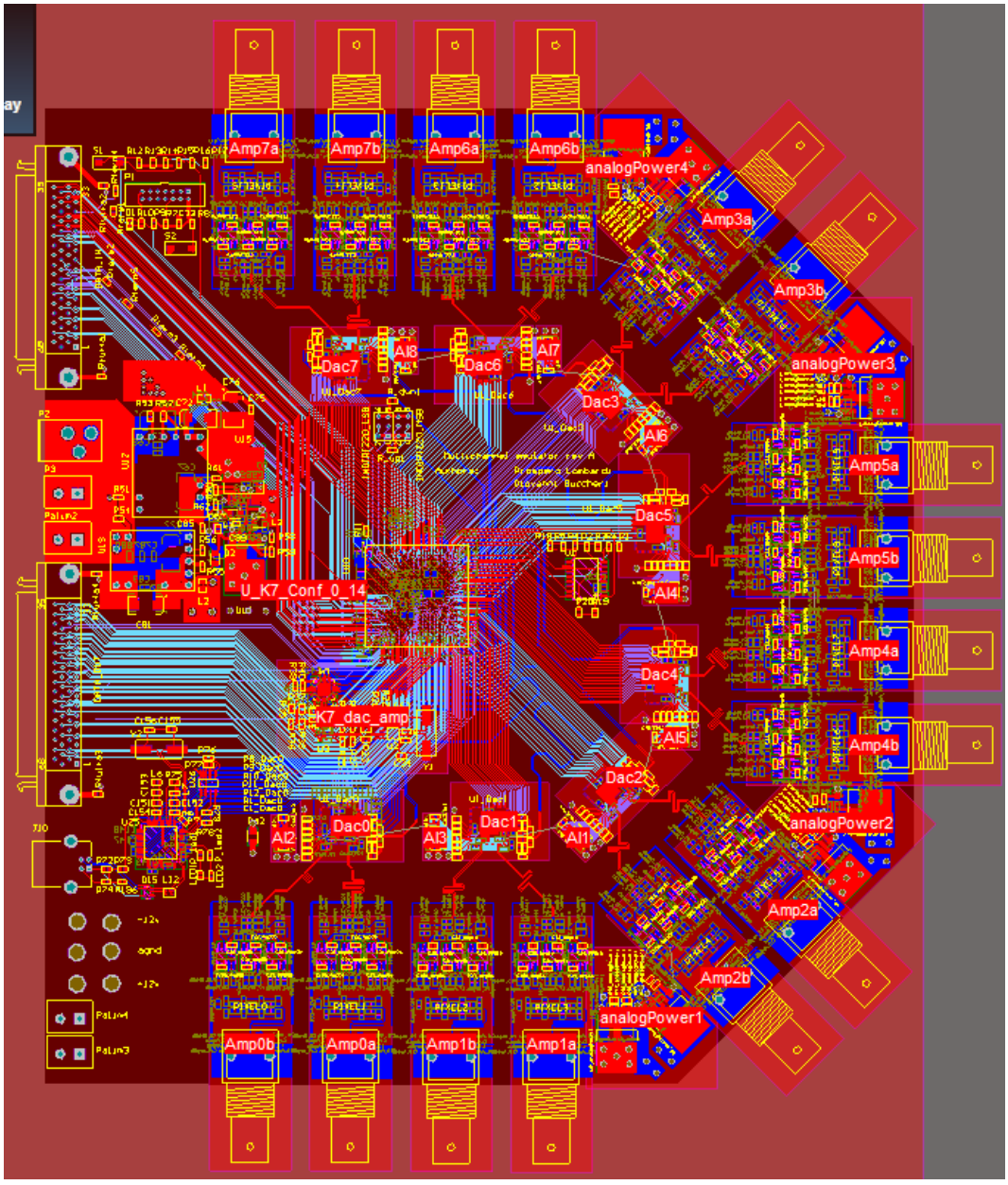

Figura 6.2: *Layout del prototipo.*

paralleli sincroni single ended con frequenza di 125MHz.

Sulla scheda sono presenti diversi sottosistemi di alimentazione. Il modulo di alimentazione digitale, posizionato sulla sinistra nella zona intermedia tra i due bus ha il compito di alimentare correttamente l'FPGA, la FLASH,

i DAC e l'USB. Quattro moduli di alimentazione analogica stabilizzano la tensione per i 16 moduli di condizionamento analogico e 8 moduli secondari regolano la tensione analogica di riferimento per i DAC.

La scheda è stata realizzata su PCB FR4 a 8 layer di cui si riporta in figura refstackup la configurazione elettrica adottata.

6.1 Integrità dei segnali

Le problematiche affrontate nello sviluppo della scheda sono quelle che tipicamente si incontrano nel layout di sistemi mixed signal. Studiare l'integrità dei segnali implica l'analisi di tutte le possibili fonti di degradazione dell'informazione in un sistema digitale. Nel seguito non si illustreranno tutte le possibili precauzioni necessarie al fine di far operare in modo corretto un sistema digitale high speed, piuttosto si evidenzieranno le due maggiori criticità affrontate nell'implementazione della scheda di emulazione.

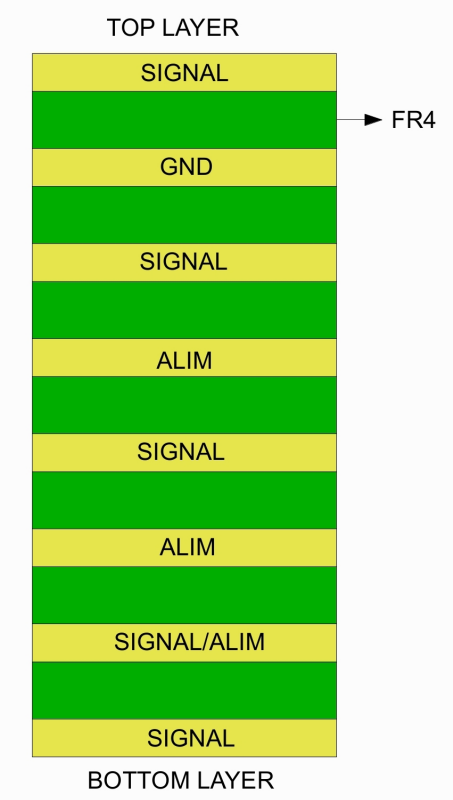

Figura 6.3: *Stackup del PCB.*

6.1.1 Riflessioni

E' noto che nel contesto dell'elettronica digitale tutti i segnali di tensione o corrente vengono interpretati come "valori logici". Una tensione che supera una certa soglia logica viene interpretata, ad esempio, come 1 altrimenti come 0. Realizzando sistemi ad alta velocità si ci scontra con dei fenomeni potenzialmente dan-

nosi per l'informazione digitale trasportata dai segnali elettrici. Lo studio dell'integrità dei segnali permette di applicare opportune strategie al fine di contrastare la corruzione dell'informazione e permettere al sistema di operare con frequenze di clock elevate.

Operando con segnali a bassa frequenza un generico conduttore di collegamento viene tipicamente modellizzato attraverso la sua resistenza. Questa rappresentazione è sicuramente inadatta nel caso in cui lo stesso conduttore venga utilizzato per veicolare segnali ad alta frequenza. In questi casi è opportuno utilizzare un approccio a parametri distribuiti modellizzando una certa linea attraverso la sua impedenza caratteristica.

Presi due conduttori affiancati, uno di andata ed uno di ritorno per il segnale, esisterà una certa capacità ed una certa induttanza per unità di lunghezza che terrà conto della reciproca vicinanza, della forma dei conduttori e del materiale dielettrico interposto fra essi. L'impedenza caratteristica è definita come il rapporto fra l'onda di tensione e quella di corrente iniettate nel sistema appena descritto e può essere calcolata attraverso la nota espressione $Z_0 = \sqrt{\frac{L}{C}}$ dove L e C sono l'induttanza e la capacità per unità di lunghezza del sistema che prende il nome di linea di trasmissione. Quando alla linea di trasmissione si collega un generatore ad un lato ed un carico R_L all'altro, si hanno riflessioni di segnale nel caso in cui risulti $R_L \neq Z_0$ (mancato matching delle impedenze). Poiché un sistema digitale discrimina valori logici alti da valori logici bassi in ragione del superamento di una certa soglia logica, il matching delle impedenze diventa critico quando in una trasmissione digitale riflessioni multiple vanno a modificare pesantemente l'ampiezza del segnale di tensione ricevuto al carico.

Nel caso di piste su PCB è importante sottolineare che esistono due tipi di linee di trasmissione; le microstrip e le stripline. Queste differiscono per il posizionamento del conduttore di andata: le prime sono tracciate in uno dei due layer esterni del PCB (top o bottom) e sono posizionate sopra un piano di massa che funge da conduttore di ritorno. Le seconde si trovano in strati interni del circuito stampato e quindi sono impacchettate tra un piano di alimentazione superiore ed uno inferiore. L'impedenza caratteristica viene in entrambi i casi controllata attraverso le proprietà geometriche delle piste

ovvero la loro distanza dal piano, la loro altezza e larghezza. In questo modo il progettista ha pieno controllo su Z_0 ed è in grado di effettuare il collegamento tra dispositivi (routing) in condizioni di matching impedenziale per evitare riflessioni.

L'adattamento di impedenza può essere realizzato in ingresso alla linea o alla sua terminazione. Nel primo caso si realizza un adattamento serie ponendo $R + R^{driver} = Z_0$ dove R^{driver}rappresenta la resistenza equivalente esibita in uscita dal circuito trasmittente e R è la resistenza che realizza il matching. Nel secondo caso è consuetudine utilizzare un resistore $R_L = Z_0$ connesso verso massa subito dopo la fine della linea oppure una coppia di resistori in modo tale che il parallelo dei due coincida con Z_0.

6.1.2 Crosstalk

La seconda problematica che affrontiamo è il crosstalk (o diafonia). Il crosstalk è un fenomeno tale per cui un segnale trasmesso in un circuito detto aggressore crea un effetto indesiderato in un altro circuito detto vittima. E' tipicamente causato da accoppiamento capacitivo ed induttivo tra aggressore e vittima.

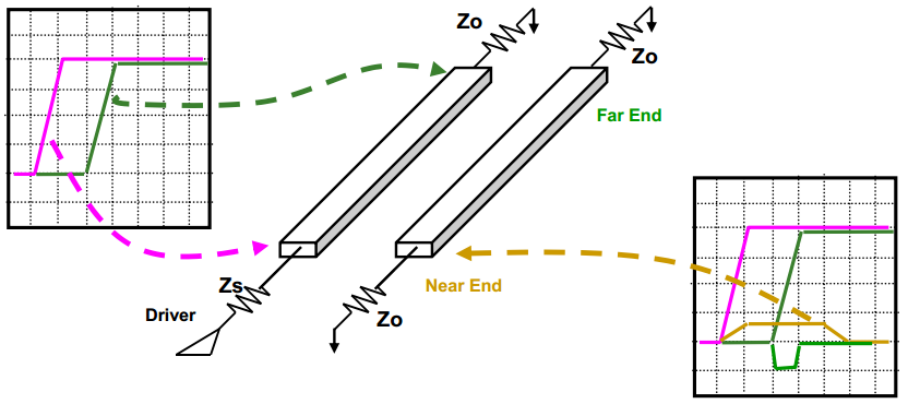

Figura 6.4: *Segnale di crosstalk tra due linee.*

Consideriamo la situazione in figura 6.4 in cui due piste siano posizionate vicine tra di loro e in cui la corrente fluisca in una delle due, ovvero nella pista aggressore, in seguito ad un gradino di tensione con pendenza finita; la

situazione viene studiata, nell'ambito della compatibilità elettromagnetica, come una interazione a campo elettromagnetico vicino.

Il risultato della sovrapposizione dei due fenomeni di interazione, induttivo e capacitivo, è la generazione di segnali molto diversi ai due estremi della linea vittima. Il segnale positivo al near end (vedi figura) ha una durata pari al doppio del tempo necessario al segnale aggressore per propagarsi lungo tutta la linea di trasmissione mentre al far end si ha un impulso negativo di durata pari al tempo di salita del segnale aggressore.

Il crosstalk, specialmente quello di natura induttiva, può diventare un problema nei sistemi digitali quando il livello dei segnali indotti è tale da corrompere l'informazione trasportata nel circuito vittima. Fortunatamente i sistemi digitali sincroni presentano una naturale immunità a tale fenomeno. Immaginiamo di osservare un semplice bus di comunicazione a n bit ove le piste di collegamento siano molto ravvicinate al fine di ridurre l'ingombro su PCB; tipicamente si hanno distanze inferiori al millimetro. Se il bus in questione è sincrono i dati vengono campionati dal circuito ricevitore in corrispondenza dei fronti del segnale di clock. Poiché i fenomeni di diafonia sono presenti soltanto in presenza di variazioni di segnale, la loro durata temporale è confinata, sulle piste del bus, all'interno della finestra di commutazione dei dati. Per questo motivo il segnale di clock, opportunamente sfasato rispetto alla commutazione dei segnali, in assenza di ulteriori fenomeni di interferenza esterna campiona i dati stabilizzati privi di spurie.

L'accoppiamento tra piste vicine è tipicamente molto più importante nell'ambito dei sistemi mixed signal in quanto l'analogica è caratterizzata da una minore immunità ai disturbi elettromagnetici. Infatti considerando di operare con un DAC con un numero di bit equivalente pari a $12bit$ e con una dinamica dell'uscita di $3V$ la minima variazione di segnale è $V^{min} = \frac{3V}{2^{12}} \sim 0,73mV$. Questo valore è molto inferiore rispetto ad un tipico segnale indotto da crosstalk nelle commutazioni digitali che può superare le centinaia di mV.

Una pratica comune è, dunque, quella di posizionare sul PCB le parti di elaborazione analogica del segnale il più lontano possibile dall'elettronica digitale.

Nella scheda i moduli di amplificazione analogica di uscita che si interfacciano con i DAC sono stati disposti al di sotto della scheda mentre tutti i dispositivi digitali si trovano al di sopra; interponendo un piano di alimentazione tra le due sezioni si è aggiunta un azione schermante che va ad unirsi a quella realizzata attraverso delle gabbie di Faraday. Ogni modulo analogico infatti viene individualmente chiuso all'interno di un involucro metallico. E' importante sottolineare che gli accorgimenti adottati non sono soltanto precauzionali ma indispensabili per avere una corretta emulazione. Una prima implementazione prototipale dello stadio di uscita (DAC+analogica) ha messo in evidenza la presenza di componenti armoniche, sovrapposte alla forma d'onda emulata, alla stessa frequenza del clock di sistema con valori anche superiori ai 150mV, che degradavano pesantemente il rapporto segnale rumore in uscita.

6.2 Distribuzione alimentazione

Una corretta alimentazione del sistema è assolutamente fondamentale per il buon funzionamento dei dispositivi elettronici. Possiamo dire che essa è anche più importante della risoluzione delle problematiche di integrità dei segnali; banalmente, se l'alimentazione è inadeguata l'intero circuito non funziona affatto. Alimentare un sistema elettronico high speed non è semplice come si potrebbe pensare.

Lo scaling dei dispositivi ha comportato per i sistemi digitali una riduzione progressiva delle tensioni di alimentazione. Valori tipici che si possono trovare in FPGA odierne spaziano da 3.3V a 1V. Dalla semplice constatazione che la potenza dissipata da questi dispositivi rimane più o meno costante o addirittura aumenta evolvendo verso nuove architetture si deduce che le correnti in gioco devono necessariamente aumentare. La commutazione simultanea di un grande numero di porte logiche può richiedere diverse decine di Ampere di corrente. Per questo motivo è possibile affermare che in un sistema digitale odierno l'energia elettromagnetica è maggiormente presente nella componente magnetica del campo. Il compito di un sistema di alimenta-

zione è quello di fornire in qualsiasi istante di funzionamento tutta la corrente richiesta dal carico mantenendo costante la tensione di alimentazione.

Per comprendere la criticità relativa alla distribuzione dell'alimentazione consideriamo di analizzare una topologia realistica di porte logiche come quella in figura **??**; la situazione modellizza l' alimentazione di un inverter, all'interno di un FPGA.

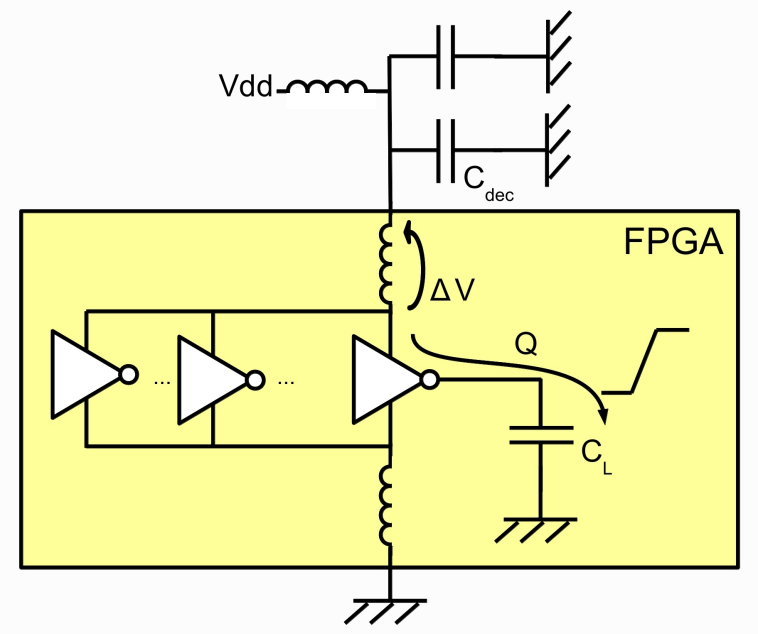

Figura 6.5: *Modellizzazione dell'alimentazione in un' FPGA.*

La presenza delle induttanze tra gli ingressi di alimentazione della porta logica e le alimentazioni stesse è giustificata dal fatto che il collegamento fisico della porta viene effettuato attraverso dei conduttori (induttanze di bonding). La porta logica, che in questo caso è un semplice inverter, viene tipicamente collegata ad altre porte logiche che possono essere modellizzate attraverso semplici capacità. Quando si ha una commutazione basso-alto dell'uscita dell'inverter, per caricare la capacità $C_L$al valore di tensione alto è necessaria una certa carica Q proveniente, naturalmente, dell'alimentazione. Se la transizione è molto veloce, circostanza auspicabile in un sistema digitale veloce, la variazione della corrente richiesta nell'unità di tempo è elevata. Per questo motivo ai capi dell'induttanza parassita L^{par} si svilupperà una

certa caduta di tensione semplicemente pari a $\triangle V = L^{par} \frac{di(t)}{dt}$. Durante la commutazione la porta logica si trova ad essere alimentata con una tensione inferiore a V_{dd} della quantità $\triangle V$. La caduta di tensione può essere tale da spegnere il circuito stesso anche se il valore dell'induttanza è relativamente basso in quanto il tempo di commutazione è molto breve. Al fine di evitare questo fenomeno di *scollegamento dell'alimentazione* viene adottata un'opportuna strategia. Inserendo una capacità tra alimentazione e massa il più possibile vicino ai pin di alimentazione del dispositivo, nel momento in cui la porta logica commuta la corrente non fluisce attraverso tutto il collegamento dell'alimentazione ma viene erogata momentaneamente dalla capacità. Questa verrà successivamente ricaricata attraverso l'alimentazione in seguito alla commutazione. Il meccanismo può essere interpretato anche in termini di filtraggio. La capacità, detta di decoupling, abbassa l'impedenza del nodo di alimentazione quando si hanno richieste di corrente ad alta frequenza. Per ridurre l'effetto negativo dell'induttanza serie ESL inevitabilmente presente nei package dei condensatori, ove possibile si pongono in parallelo due o più condensatori.

Un discorso analogo vale nel caso di commutazione alto basso dell'inverter; il fenomeno prende il nome di *ground bounce* e comporta un variazione positiva del livello di massa se simultaneamente molte porte commutano verso il valore logico basso.

La scheda realizzata prevede condensatori di decoupling per tutti i dispositivi impiegati ed in particolar modo per l'FPGA; oltre una trentina di condensatori sono stati posizionati al di sotto di questa. L'FPGA ha un grande numero di pin dedicati all'alimentazione ad ognuno dei quali è collegata una piccola capacità di decoupling molto veloce internamente al package. Si sono pertanto posizionati una ventina di condensatori da $4,7nF$, una decina da $47nF$ e la restante parte da $1\mu F$.

Il livelli di tensione richiesti sono 1V per il core dell'FPGA, 1.8V e 3.3V per i banchi di I/O, forniti attraverso due piani di alimentazione in virtù di un ottimale raggruppamento dei pin.

La stabilizzazione delle tensioni è affidata a tre regolatori. La tensione di alimentazione primaria della scheda di 12V viene convertita ad 1V attraverso

un regolatore switching PTH08T220W. Analogamente la tensione 3.3V viene generata attraverso un PTH08T240W. Da questo livello di tensione viene poi generata la 1.8V.

Le tensioni 3.3V e 1.8V vengono distribuite al resto dell'elettronica digitale insieme al livello 2.5V, generato da un regolatore lineare aggiuntivo, richiesto dallo stadio di generazione e distribuzione del clock.

Per fare in modo che le correnti di ritorno prodotte durante il funzionamento del sistema fossero contenute all'interno della zona digitale della scheda, il piano di massa digitale ed il piano di massa analogica sono state collegate in un solo punto ed inoltre tutti i circuiti relativi all'alimentazione digitale sono stati posizionati il più possibile lontano dall'elettronica analogica al fine di minimizzare i disturbi indotti.

L'alimentazione analogica a differenza di quella digitale è stata partizionata su più moduli al fine di rendere omogenea la dissipazione del calore. Un bus di alimentazione fornisce i livelli di tensione +12V, -12V e massa analogica ai 4 moduli di alimentazione, delocalizzati sul perimetro della scheda, ognuno dei quali fornisce l'alimentazione stabilizzata di +10V e -10V a 4 canali. Allo stesso bus di alimentazione vengono collegati i moduli per l'alimentazione analogica 3,3V dei DAC.

6.3 Comunicazione

6.3.1 Segnali differenziali

Un segnale differenziale rappresenta un valore come la differenza tra due livelli fisici. In un certo senso tutti i segnali di tensione sono differenziali poiché essi vengono misurati rispetto ad una tensione di riferimento. In molti sistemi questa tensione di riferimento è quella di massa; i segnali che hanno come tensione di riferimento quella di massa vengono detti single ended. Questa dicitura deriva dal fatto che i segnali vengono rappresentati attraverso tensioni che si sviluppano su un unico conduttore. Al contrario un segnale differenziale si sviluppa su due conduttori. Propriamente il valore di segnale è dato dalla differenza di potenziale esistente tra i due conduttori etichettati

come V^+e V^-. Il segnale è definito positivo se la tensione sul conduttore V^+è superiore a quella sul conduttore V^-e negativo nel caso duale. La tensione media attorno alla quale il potenziale dei due conduttori si muove viene detta tensione di *modo comune* del segnale differenziale. Utilizzando l'approccio differenziale, ogni collegamento richiede una complessità doppia rispetto al caso single ended per poter essere realizzato. A fronte di un aumento di complessità esistono vantaggi indiscussi che giustificano l'utilizzo dei segnali differenziali altrimenti non convenienti dal punto di vista dell'ingombro sul circuito stampato. Un sistema elettronico costituito da dispositivi interconnessi mediante collegamenti single ended spesso possiede un unico riferimento di massa. Siccome il riferimento di massa non è mai uguale per tutti i dispositivi (basti pensare al fenomeno del ground bounce o a cadute di tensione sul piano di massa), i segnali single ended ai ricevitori possono differire notevolmente da quelli trasmessi. In un sistema differenziale, invece, il recupero del valore di un segnale è indipendente, entro certi limiti, dall'esatto valore della massa.

Il secondo grosso beneficio consiste nell'elevata immunità ai disturbi elettromagnetici. Poiché le due piste differenziali vengono tracciate molto vicine, un eventuale disturbo esterno manifesta su entrambe gli stessi effetti. Tipicamente si ha un'alterazione della componente di modo comune che come abbiamo visto non ha effetto sul contenuto informativo trasportato dal sistema. Per una nota proprietà di compatibilità elettromagnetica se un circuito è poco suscettibile ad interferenza allora esso sarà anche caratterizzato da basse emissioni. Questa circostanza è molto utile nel caso di collegamenti lunghi e con piste vicine.

Inoltre un sistema differenziale fornisce il doppio della dinamica rispetto ad uno single ended presa una certa escursione di segnale. Per esempio, considerata una tensione di alimentazione di $+5V$, un segnale single ended può muoversi entro una finestra limitata da $1V$a $4V$. Un sistema differenziale invece, fissato un modo comune di $2,5V$, ha un range effettivo di $\pm 3V$ (ovvero $V^+ = 4V$ e $V^- = 1V$ e poi scambiando $V^+ = 1V$ e $V^- = 4V$).

6.3.2 Standard LVDS

La tecnologia Low Voltage Differential Signaling utilizza una trasmissione differenziale dei segnali. Essa non è dipendente da una particolare tensione di alimentazione, il che vuol dire che lo standard si adatta bene a diminuzioni delle tensioni di alimentazione dovute all'evoluzione tecnologica dei dispositivi. L'LVDS garantisce una banda intorno al Gbps in un mezzo trasmissivo senza perdite; in ogni caso il rate realizzabile dipende fortemente dal canale di comunicazione impiegato. Il mezzo trasmissivo deve essere terminato con la sua impedenza caratteristica per prevenire riflessioni, che tipicamente si aggira intorno ai $100-120\Omega$. In effetti il resistore di terminazione è indispensabile per generare la tensione appropriata in ingresso al ricevitore poiché il driver ha un'uscita in corrente.

I livelli di tensione tipici dello standard sono $+1,2V$ per il modo comune e di $\pm 400mV$ per il modo differenziale.

Al fine di evitare che un disturbo di modo comune possa produrre un segnale differenziale spurio al ricevitore è necessario che la lunghezza delle linee differenziali sia il più possibile uguale.

Oltre allo standard LVDS nel sistema si è fatto uso anche dello standard differenziale LVPECL. Per esso viene stabilito un livello del modo comune pari a $2V$ e di $\pm 800mV$ per quello differenziale.

6.3.3 BUS interscheda

I bus che consentono la comunicazione fra le varie schede sono di tipo parallelo e sfruttano il supporto hardware dello standard SCSI a 68 poli. In particolare, del connettore standard, vengono utilizzate 16 coppie differenziali per il trasporto dei dati di emulazione, 4 per l'indirizzamento della scheda, 1 per il clock, e 6 linee single ended per segnali di controllo. Su questa interfaccia fisica viene utilizzato lo standard elettrico LVDS per il trasporto dei segnali. La presenza di bit di indirizzamento rende molto semplice la comunicazione. Ogni bus è di tipo unidirezionale da punto a punto ed ogni scheda include come accennato un collegamento in ingresso ed uno in uscita; l'interconnessione globale è illustrata in figura 6.6.

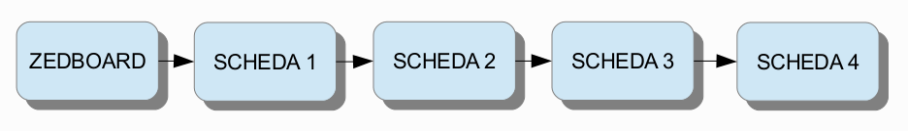

Figura 6.6: *Interconnessione reale tra le schede.*

Il cavo utilizzato presenta una impedenza caratteristica di 110Ω cioè rientrante nel range compatibile con lo standard LVDS. La terminazione del bus viene realizzata attraverso resistenze di terminazione a 100Ω interne all'FPGA; si viene, dunque, a creare un minimo disadattamento di $\Gamma = \frac{R^{term}-Z_0}{R^{term}+Z_0} = -0,048$. La minima frequenza di clock del bus è 64Mhz derivante dalla necessità di sostenere un rate finale in uscita intorno ad 1M eventi al secondo. Infatti considerando che la matrice di carica fornita dal position emulator contiene 64 parole a 16 bit e che ad ogni transazione di bus si può trasmettere una sola parola per volta da 16 bit, ne deriva che per trasmettere un' intera matrice di carica sono necessarie 64 transizioni sul bus.

Particolare attenzione si è dedicata al routing delle piste differenziali. In particolare si è cercato di mantenere il più basso possibile il mismatch tra i due fili componenti la coppia differenziale. Inoltre è stato regolato il rapporto di forma del sistema al fine di controllarne l'impedenza caratteristica sia nel caso di microstrip che di stripline.

6.3.4 Distribuzione del clock

In un sistema digitale sincrono il segnale di clock serve per definire la successione temporale delle operazioni di elaborazione. La rete di distribuzione del clock ha il compito di fornire il segnale di clock a tutti i dispositivi che ne hanno bisogno. Poiché la sua funzione è vitale per il sistema, è necessario prestare particolare attenzione alle caratteristiche del segnale di clock e alla rete elettrica che ha il compito di distribuirlo. I segnali di clock operano a frequenza più elevata rispetto a qualsiasi altro segnale all'interno del sistema. Siccome i segnali sono forniti in relazione all'istante temporale di commutazione del clock, è necessario che questo abbia le migliori caratteri-

stiche possibili in termini di forma d'onda e tempo di arrivo; differenze tra istanti temporali di commutazione possono inficiare le performance dell'intero sistema.

La rete di clock realizzata si compone di diversi elementi tra cui: un oscillatore al quarzo con frequenza centrale di $25MHz$, PLL, distributori di clock e linee di trasmissione.

La generazione del segnale di clock alla frequenza desiderata di $125MHz$ viene realizzata attraverso l'architettura di figura 6.7 integrata nel chip CY2XP22 della Cypress. Un circuito PLL (Phase Locked Loop) viene pilotato dal cristallo risonatore esterno e fornisce attraverso un buffer LVPECL il segnale differenziale di clock.

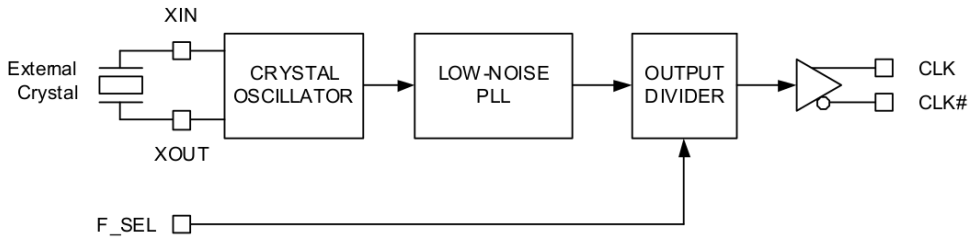

Figura 6.7: *Architettura del generatore di clock.*

Per fornire il clock all'FPGA e agli otto DAC sono stati utilizzati dei distributori di clock al fine di splittare il segnale proveniente dal generatore. In particolare è stato utilizzato prima un distributore a 2 uscite: una di queste è riservata all'FPGA mentre l'altra viene collegata ad un distributore ad 8 uscite per i DAC. La soluzione mostrata fa in modo che in partenza dal distributore ad 8 uscite tutti i segnali di clock abbiano la stessa relazione di fase. Il clock dell'FPGA invece ha un anticipo di circa 450ps determinato dal ritardo di propagazione tipico del distributore. Facciamo notare che la soluzione ingenua di collegare tutti i dispositivi attraverso una struttura ad albero è assolutamente impraticabile a causa delle inevitabili riflessioni che si vengono a creare nonostante si adottino strategie di matching impedenziale.

Una delle maggiori problematiche che si incontra nella realizzazione della rete di clock è l'equalizzazione delle lunghezze delle piste. Siccome su un PCB FR4 la propagazione di un segnale può subire ritardi che vanno dai

$120ps/in$ nel caso di microstrip fino ai $180ps/in$ nel caso di stripline, è molto semplice ottenere un ritardo sistematico nell'arrivo dei fronti (skew). Se due componenti ricevono il clock di sistema in instanti diversi c'è il rischio di incombere in violazioni dei tempi di setup o di hold tipici dei dispositivi.

Facciamo una semplice considerazione. Nella scheda realizzata il piazzamento dei componenti è tale che la differenza di distanza, tracciando il collegamento più corto possibile tra il componente più lontano e quello più vicino al distributore di clock, è di circa $10in = 25,4cm$. Questo comporterebbe un ritardo fra i due segnali di clock di almeno $1,2ns$ il che è di fatto intollerabile visto che il periodo del segnale di clock è di soli $T^{period} = \frac{1}{125MHz} = 8ns$. La strategia adottata al fine di ottenere una buona distribuzione del segnale è stata quella di posizionare le linee di trasmissione sul layer bottom, offrendo questo il minor ritardo di propagazione, in una configurazione a stella. Tutte le linee ad impedenza controllata di 100Ω vengono tracciate dal distributore fino al centro del sistema e successivamente diramate fino a collegare ciascun dispositivo. Il massimo mismatch che si è ottenuto dopo una successiva e più fine equalizzazione è di soli $100mils = 2,5mm$.

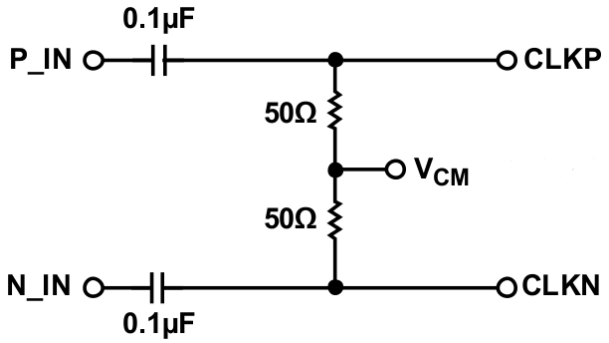

Figura 6.8: *Rete di traslazione e adattamento dei livelli. A sinistra viene collegato il segnale LVPECL ed a destra abbiamo il segnale in uscita.*

Facciamo notare che siccome la distribuzione del clock segue lo standard LVPECL è stato necessario utilizzare la rete di traslazione ed adattamento dei livelli che si riporta in figura 6.8. Ogni DAC e l'FPGA sono muniti della rete passiva che oltre ad effettuare la giusta conversione dei segnali termina efficacemente la linea di trasmissione.

6.4 Condizionamento analogico di uscita

I convertitori DAC utilizzati vengono configurati attraverso un semplice bus SPI per abilitare la modalità interleaved. In questo modo è possibile sfruttare un solo bus di comunicazione tra l'FPGA ed il singolo DAC per fornire i campioni relativi ad una coppia di canali. Il massimo sample rate che il dispositivo può sostenere è di $250 Msample/s$, in modalità interleaved i dati vengono campionati sia sul fronte in discesa che su quello in salita del clock.

Il segnale di uscita del convertitore è un sistema differenziale in corrente. Al fine di convertire il segnale generato in un segnale di tensione viene utilizzato uno stadio di amplificazione per strumentazione INA il cui schematico viene riportato in figura 6.10. In cascata all'amplificatore e subito prima del connettore coassiale con impedenza caratteristica canonica di 50Ω viene posizionato un filtro risonante LC a 3 celle. In particolare é stato scelto un filtro di Bessel passa basso del settimo ordine con banda passante di $90 MHz$ al fine di ottenere un giusto compromesso tra distorsione dell'ampiezza di segnale in banda e sfasamento delle componenti armoniche. Il filtri di Bessel, infatti, sono caratterizzati da una risposta in fase la più lineare possibile in banda passante. Questa caratteristica fa si che il filtro introduca un ritardo temporale uguale per tutti i segnali con frequenza inferiore al taglio.

La frequenza di cut off è stata scelta in modo tale da non filtrare eccessivamente le componenti armoniche del segnale ed allo stesso tempo da avere una buona attenuazione di possibili componenti alla frequenza del clock di $125 MHz$. Con questa configurazione si ottiene un'attenuazione di circa $27 dB$ alla frequenza del clock.

In figura 6.9 si riporta la simulazione circuitale dello stadio analogico su grafico bilogaritmico. Lo stadio viene stimolato attraverso l'equivalente Norton del convertitore DAC e fornisce una dinamica in uscita di circa $\pm 1,8 V$. In particolare si fa notare che, siccome il DAC presenta un'uscita in corrente ad elevata impedenza, vengono riportate soltanto le due resistenze da 50Ω in ingresso all'INA che effettuano la conversione corrente-tensione.

Il segnale del DAC è caratterizzato da una componente di modo comune e da una componente differenziale entrambe di intensità $10 mA$. Quando in

ingresso al DAC viene asserita la parola digitale $0xFFFF$ vengono erogati $20mA$ da un terminale di uscita e $0mA$ dall'altra mentre nel caso di parola $0x0000$ le due correnti si scambiano. Nella simulazione è stata applicata in ingresso una sinusoide a frequenza variabile da $10MHz$ ad $1GHz$ al fine di osservare l'azione passa basso del filtro. La curva a minor pendenza tiene conto del semplice filtraggio RC in ingresso e di quello dell'operazionale terminale dello stadio mentre la curva a maggior pendenza è stata derivata al nodo di uscita dopo il filtro di Bessel.

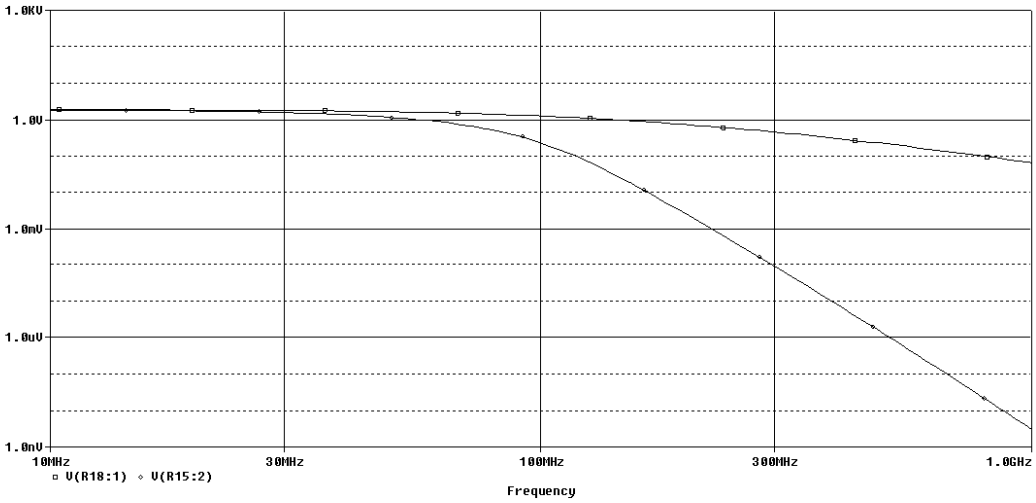

Figura 6.9: *Risposta in frequenza dello stadio analogico.*

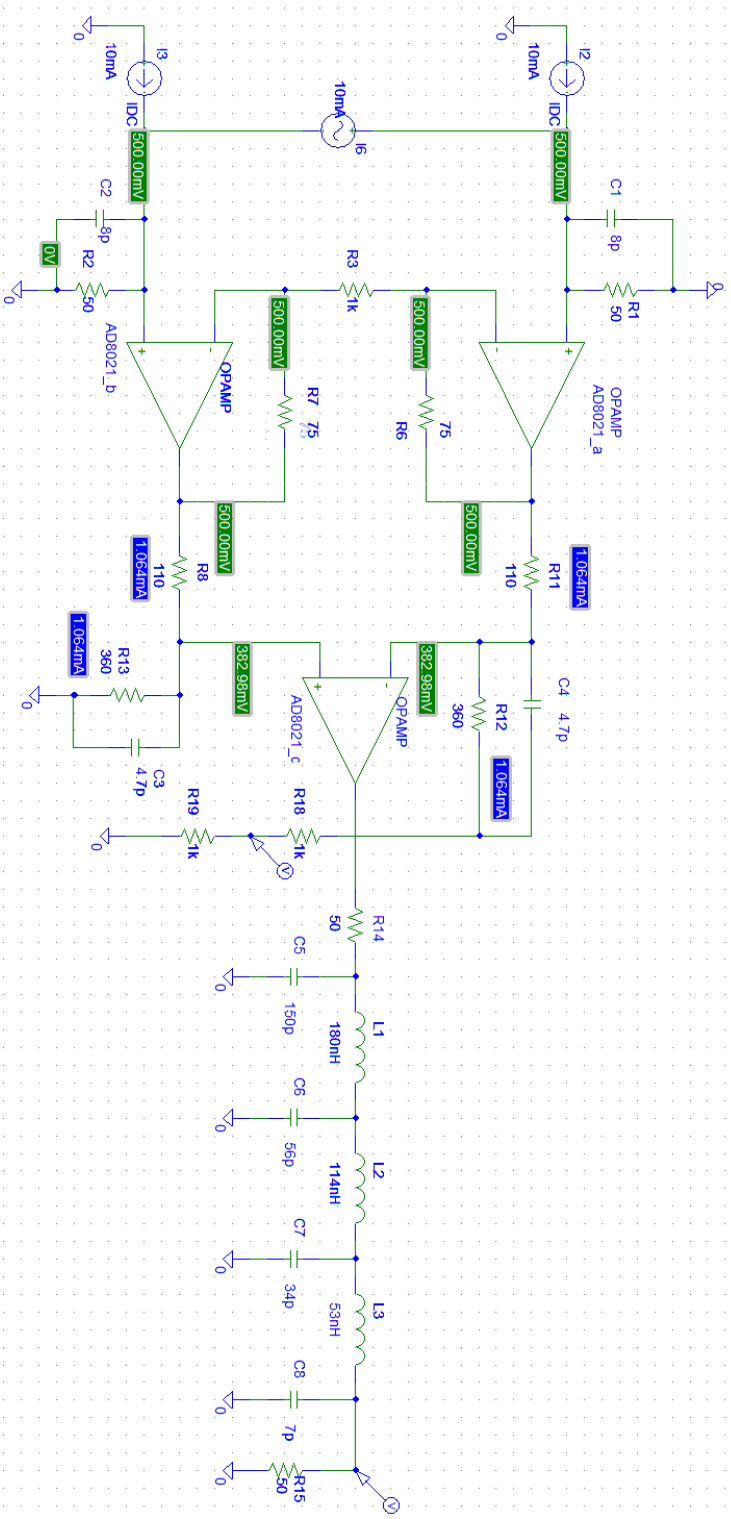

Figura 6.10: *Schema circuitale dello stadio analogico di uscita.*

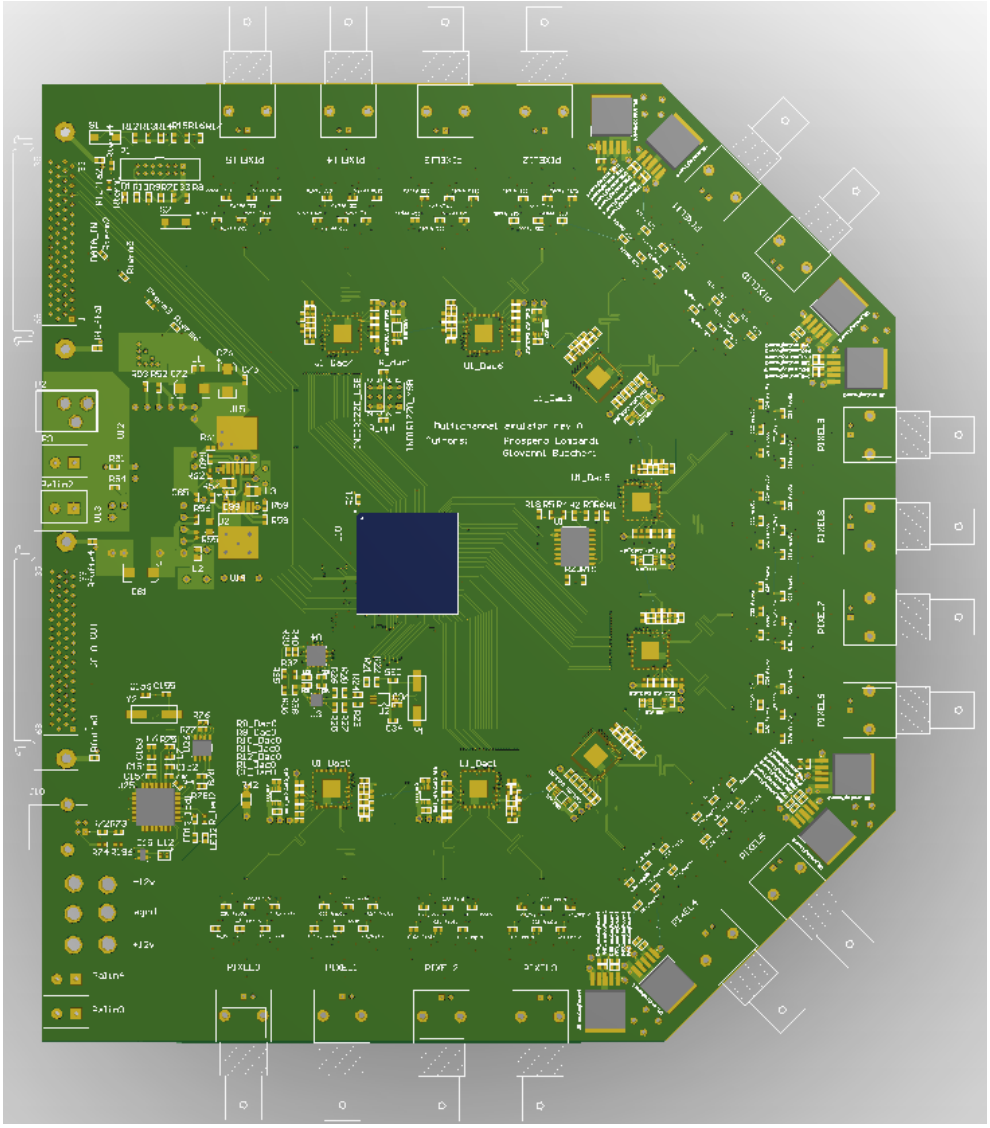

Figura 6.11: *Vista 3D top della scheda.*

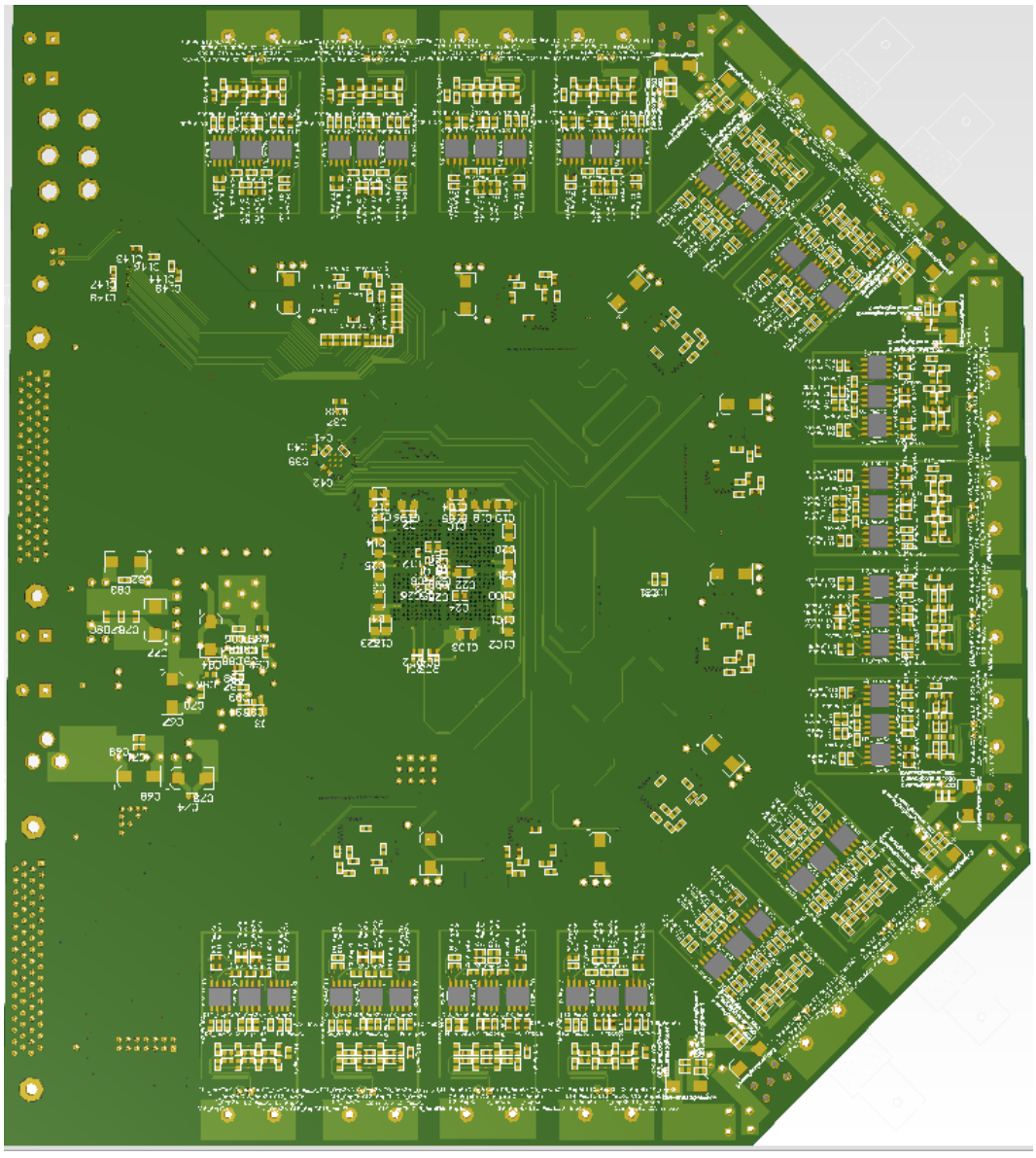

Figura 6.12: *Vista 3D bottom della scheda.*

Capitolo 7

Misure e conclusioni

Il lavoro ad oggi sviluppato deve essere inteso come un'evoluzione di un emulatore a doppio canale già realizzato presso il gruppo di ricerca ove l'attività si è svolta. Questo dispositivo permette l'emulazione di due singole sorgenti radioattive aventi proprietà del tutto analoghe a quelle analizzate nel caso della radiazione gamma per scopi di diagnostica nucleare.

La distribuzione delle energie e la statistica di emissione temporale sono sicuramente problematiche comuni, così come pure l'esigenza di dover ricostruire con esattezza la forma d'onda del segnale in uscita ed il relativo rumore. L'elaborazione, interamente effettuata su FPGA, è stata già ampiamente testata ed ha mostrato un'elevata fedeltà di emulazione. Basandosi sull'implementazione dell'emulatore a doppio canale, la realizzazione di quello multicanale ha richiesto, di fatto, l'estensione dell'emulazione a livello matriciale.

Riferendosi all'emulazione puntiforme, realizzata dall'emulatore a doppio canale, non ha senso parlare della probabilità che un fotone abbia inciso in una particolare posizione dello scintillatore poiché, in linea di principio, il singolo detector accoppiato produce sempre la stessa quantità di carica in uscita. E' però vero che la quantità di carica e, dunque, l'ampiezza dell'impulso in uscita allo stadio di preamplificazione, dipendono dall'energia del fotone incidente. Anche la distanza temporale tra impulsi successivi segue la statistica Poissoniana come nel caso dell'emulazione multicanale.

La difficoltà nello sviluppo del progetto è stata, dunque, quella di testare

un modulo capace di produrre con efficacia una statistica spaziale di emissione. Ciò è stato realizzato come abbiamo visto attraverso l'emulatore della posizione in quanto i blocchi successivi, se pur con leggere modifiche, possono essere riprodotti basandosi sull'emulatore a doppio canale.

Per testare l'efficacia dell'emulazione matriciale si è proceduto lanciando un'esecuzione del firmware sviluppato per lo Zynq. Due sono le criticità che è stato necessario verificare:

- I valori prodotti dal blocco dovevano riprodurre la statistica dell'immagine;
- I valori dovevano essere prodotti e trasmessi dal deliverer in tempo reale ovvero ad un rate di circa 1M eventi al secondo.

Si è proceduto su due fronti diversi: per verificare il primo obiettivo si è intervenuti sul codice C e per l'altro sia sul codice C che sul codice VHDL da implementarsi sulla parte FPGA dello Zynq.

Per testare la correttezza dei dati, si è lanciata un'esecuzione di una versione lievemente modificata del firmware al fine di collezionare un buon numero di matrici di carica. Per il buffer condiviso tra i due processi è stata scelta una dimensione di 512MB poiché una maggior grandezza avrebbe intaccato le prestazioni del sistema operativo. In questo modo è stato possibile salvare temporaneamente in memoria 4 milioni di matrici charge. Il firmware dopo l'emulazione vera e propria salva in un file binario il contenuto del buffer condiviso in modo tale da poter effettuare debug tramite il programma di calcolo Matlab. In particolare attraverso l'algoritmo del baricentro si è effettuata una ricostruzione dell'immagine di partenza il cui risultato è visibile in figura 7.1. L'emulazione produce lo stesso risultato della simulazione Matlab. Visto il rate che ci si aspetta di ottenere l'esecuzione ha impiegato un tempo inferiore ai 5 secondi per essere portata a termine. Questa misura del tempo è stata effettuata attraverso un meccanismo automatico fornito dal sistema operativo Linux basato sul conteggio dei colpi di clock del sistema.

Al fine di avere una maggior precisione nella valutazione del rate di generazione sostenibile dall'emulazione si è sviluppato un codice di controllo

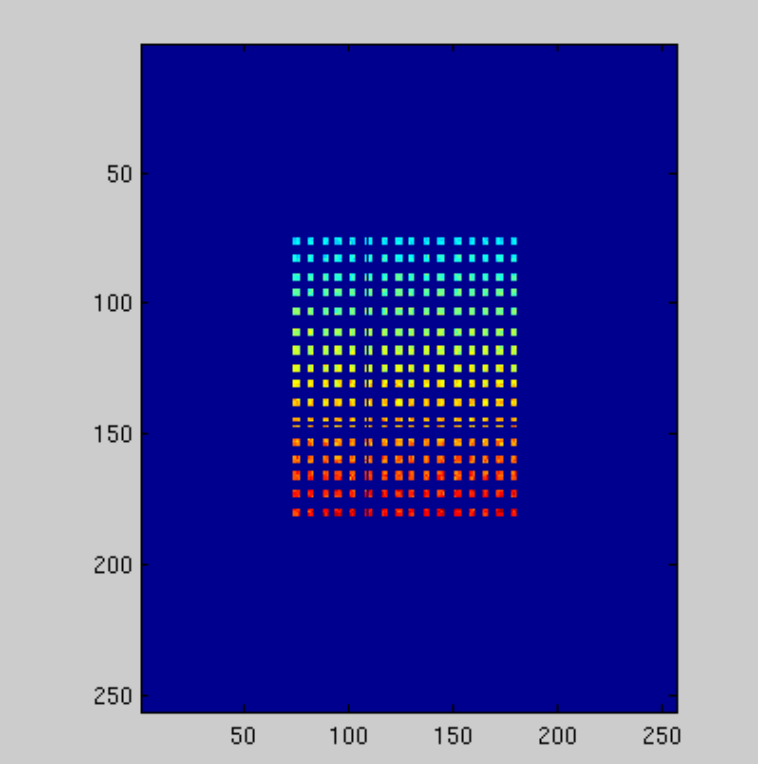

Figura 7.1: *Risultato dell'emulazione.*

in VHDL. Il modulo di controllo è stato interfacciato alla BRAM interna al deliverer così da osservare eventuali latenze o sovraccarichi del sistema.

Il modulo di controllo effettua una semplice verifica dei dati scansionando in sequenza tutte le celle della memoria. Ad un colpo di clock viene letta la cella n contenete il valore $Data_n$ mentre a quello precedente era stata letta la cella n-1 contenente $Data_{n-1}$. Se la condizione:

$$Data_n = Data_{n-1} + 1$$

non si verifica allora c'è stato un errore. In questo caso si azzera un contatore che tiene traccia della distanza, espressa in numero di colpi di clock, tra due errori consecutivi. Se si trasmettessero nella BRAM i valori di emulazione allora ad ogni colpo di clock ci sarebbe un errore visto che i dati non rispettano sicuramente tale condizione. Il codice in esecuzione sul processore è stato, allora, modificato in modo tale che nel buffer condiviso

non venissero copiati i risultati delle singole emulazioni ma blocchi di valori rappresentanti una rampa digitale. Facciamo notare che questa modifica non altera il tempo di esecuzione poiché si ha soltanto un diverso vettore di origine dei dati trasferiti nel buffer condiviso; invece che utilizzare il vettore contenente le $PSF*$ si ricorre al vettore rampa e tutte le altre parti del codice rimangono identiche.

Il risultato di questa modifica è che alla BRAM viene trasmessa una rampa crescente con un numero totale di campioni tre volte superiore alla sua dimensione. Una volta raggiunto il valore massimo della rampa si passa allo zero e la rampa ricomincia; questa transizione si ha sempre in corrispondenza del passaggio dall'ultima cella della BRAM alla prima e non si genera mai un errore.

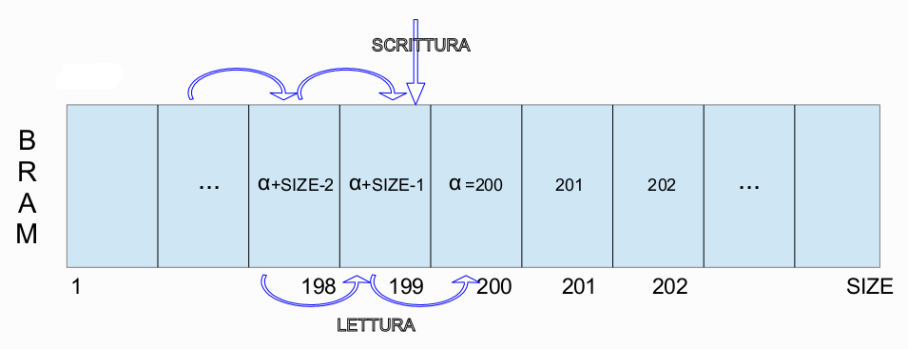

Figura 7.2: *Riempimento della BRAM.*

Analizziamo quello che succede a regime aiutandoci con la figura 7.2 dove viene rappresentata la BRAM. In alto è raffigurata la scrittura di nuovi valori ed in basso la lettura degli stessi da parte del blocco di controllo, entrambe effettuate da sinistra a destra ovvero secondo gli indirizzi crescenti. Alla scrittura della prima parte della rampa la cella 200 conterrà il valore 200 della rampa, la cella 201 il valore 201 e così via. La seconda parte della rampa andrà a sovrascrivere i valori precedentemente salvati nelle celle ed in particolare si avrà che la cella 198 conterrà il valore 200+SIZE-2 invece che 198 e quella 199 il valore 200+SIZE-1 invece che 199 dove SIZE indica il numero totale di celle della BRAM. In questo modo si viene a creare una discontinuità dei dati facilmente rilevabile dal blocco di controllo. Siccome

la velocità di lettura f^{read} è nota ed è scelta maggiore di quella di scrittura, il blocco di controllo doppierà di continuo la scrittura. In questo modo il calcolo della velocità di scrittura f^{write} è molto semplice infatti risulta:

$$f^{write} = f^{read} \frac{counter - SIZE}{counter}$$

dove counter è il numero di colpi di clock che separano due errori. In effetti ogni volta che si ha un errore si misura una velocità media di scrittura di esattamente $counter - SIZE$ valori.

Le misure effettuate tramite questa procedura hanno mostrato che il sistema riesce a sostenere un tasso medio di eventi di 910Mcount al secondo.

L'emulatore della posizione è, ad oggi, il collo di bottiglia del sistema in quanto i blocchi implementati su FPGA riescono a sostenere rate di eventi ben più alti. Un miglioramento delle prestazioni dovrà essere sicuramente focalizzato su una parallelizzazione più spinta del codice effettuandone, ad esempio, il porting verso architetture ARM quad core.

Bibliografia

[1] Andrea Abba "Methodology and advanced electronic achitectures for high performance digital processing"; doctoral dissertation;

[2] Emilio Matricciani "Lezioni di probabilità e processi aleatori"; Progetto Leonardo 2009;

[3] Guido Valli, Giuseppe Coppini "Bioimmagini" ; Patron 2005;

[4] P.Brusca, R.Peloso, A.Gola, A.Abba "The HICAM gamma camera";

[5] P.Brusca, R.Peloso, A.Gola, A.Abba "Applications of the HICAM gamma camera".

[6] Sergio Cova, M. Bertolaccini "Tubi a vuoto e fotorivelatori";

[7] Jonathan Corbet "Linux device driver" III edizione;