Università degli Studi di Roma "La Sapienza"
Facoltà di Scienze Matematiche e Naturali
Corso di Laurea in Fisica
Anno Accademico 2002/2003

Studio del sistema di test e controllo delle camere per i muoni dell'esperimento LHCb

Relatore interno:

Giovanni Vittorio Pallottino

Il Laureando:

Francesco Iacoangeli

N° matricola 11108277

Relatori esterni:

Valerio Bocci.

Giuseppe Martellotti

TESINE

"Misure di ozono totale tramite spettrofotometria Solare"

Relatore: A.M.Siani

"Entropia e informazione"

Relatore: M.Falcioni

Ai miei genitori

Sommario

INTRODUZIONE

Il lavoro di tesi tratta dell'esperimenti LHCb e del suo rivelatore, con un attenzione particolare al sistema elettronico di controllo dell'apparato (ECS) e al test delle camere MPWC (Multi Wire Proportional Chamber) che costituiscono la tecnologia principale del rivelatore di muoni.

LHCb è un esperimento interamente rivolto allo studio della violazione del CP nei decadimenti dei mesoni B ($b\bar{b}$) attraverso l'esame dei muoni prodotti nel processo.

Nei primi due capitoli si descrive brevemente il collisore di protoni LHC (Large Hadron Collider), in realizzazione al CERN, dove verrà effettuato l'esperimento, evidenziandone le proprietà fisiche dei fasci di particelle prodotti e come queste vengono sfruttate dal rivelatore di LHCb. Si descrive, inoltre, brevemente la struttura del rivelatore monobraccio, le sue caratteristiche e la tecnologie usate per i numerosi rivelatori che studiano le traccie lasciate dal passaggio di particelle ionizzanti, in particolare di muoni ad alta energia.

Nel terzo capitolo viene introdotto il rivelatore di muoni, la sua architettura e i processi di acquisizione dati.

Nel quarto si completa la descrizione dell'apparato sperimentale individuando l'elettronica di front-end e le richieste principali che deve soddisfare in quanto a prestazioni, costi e tolleranza alle radiazioni ionizzanti.

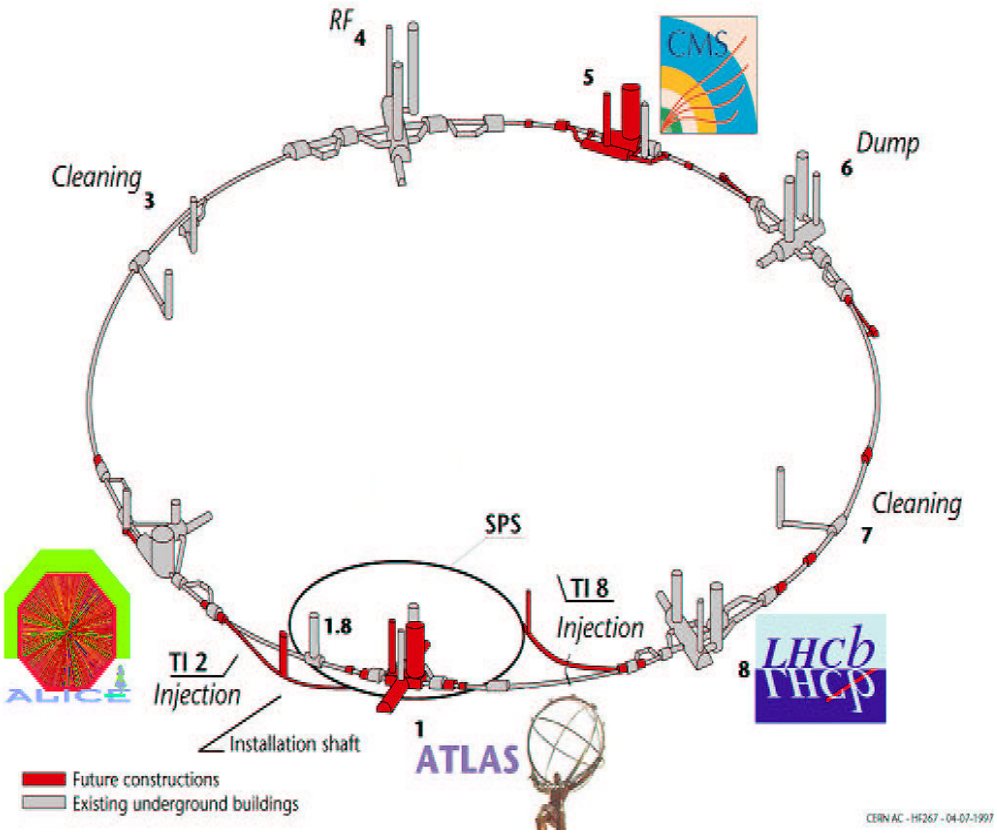
Il quinto capitolo descrive in maniera più particolareggiata architettura, tecnologia e standard dell'elettronica su cui si è svolto parte del lavoro di tesi. Vengono poi messe in luce le funzionalità implementate durante il periodo di tesi sulla Service Board, il loro utilizzo e il loro test, il test dell'elettronica con cui interagiscono.

Il sesto capitolo invece introduce il problema della caratterizzazione dell'efficienza e del jitter delle camere MWPC del rivelatore di muoni e la stazione di test ideata allo scopo. Evidenzia anche la necessità di realizzare un multiplexer per consentire lo studio dei canali delle camere con un solo TDC (time of digital converter), riducendo notevolmente i costi di realizzazione.

Infine il settimo capitolo riporta le scelte tecniche effettuate per la progettazione del multiplexer e ne descrive le fasi di disegno ed il test riportandone i risultati.

1. LHC

Figura 1.1 *LHC.*



LHC (Large Hadron Collider) è un grande collisore di protoni, formato da due anelli circolari in cui viaggiano i fasci, situato al CERN nel tunnel circolare di 8,4 km di diametro che ospitava il LEP, che sarà completato nel 2007.

Le principali caratteristiche di LHC sono riportate nella tabella sottostante (Tabella 1.1) insieme a quelle del Tevatron, il collisore protoni-antiprotoni del Fermilab, già in funzione da diversi anni.

	LHC	Tevatron
particelle collidenti	pp	$p\bar{p}$
energia nel CM	14 TeV	2 TeV
luminosità	$10^{34}\text{cm}^{-2}\text{s}^{-1}$	$2 \times 10^{32}\text{cm}^{-2}\text{s}^{-1}$
tempo tra 2 collisioni	25 ns	132 ns
circonferenza	26.7 km	6.28 km
sez. d'urto anelastica	80 mb	50 mb
sez. d'urto bb	500 μb	100 μb
rapporto bb/anelastica	6×10^{-3}	2×10^{-3}

Tabella 1.1: principali caratteristiche di LHC e di Tevatron.

Per raggiungere un alta energia nel centro di massa ($\sqrt{s} = 14\,TeV$) si utilizza lo stesso apparato d'iniezione usato per il LEP: i protoni vengono prima accelerati tramite il Protosincrotrone e il Superprotosincrotone fino ad un'energia di 0,45 TeV, poi iniettati negli anelli dove le cavità a radiofrequenza poste lungo la traiettoria forniscono l'energia mancante per arrivare ai 7 TeV per fascio.

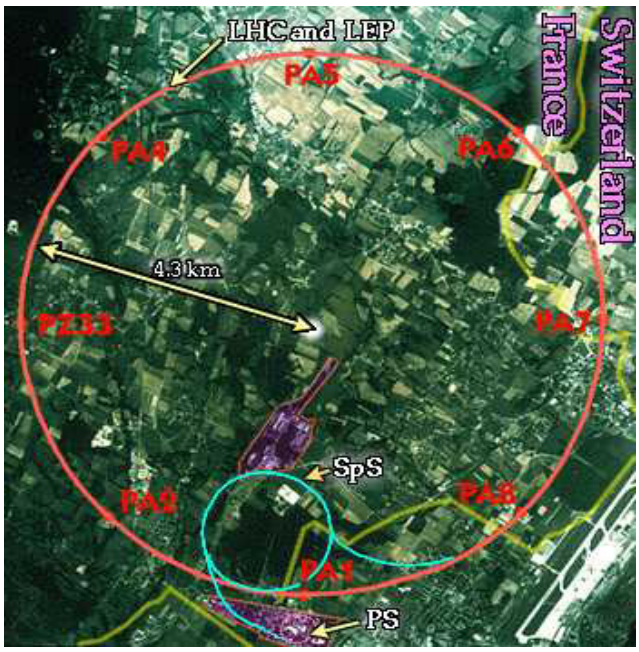


Figura 1.2: *Schema d'iniezione.*

Un ruolo importante è svolto da grandi magneti superconduttori che forniscono un campo magnetico di 8,3 tesla che permette di mantenere i fasci in un orbita circolare.
Il periodo di bunch crossing (BX), cioè la distanza temporale che intercorre tra una collisione dei fasci di particelle e la successiva, è di 25 ns.
La luminosità si può scrivere come:

$$L = F \times \frac{f n_1 n_2}{4\pi\sigma_x\sigma_y}$$

dove F=0,9 è un fattore che tiene conto dell'angolo tra i due fasci, non esattamente paralleli, n_1 e n_2 il numero di protoni, $\sigma_x\sigma_y \cong 15\mu m$ le semidispersioni gaussiane del pacchetto nella direzione di curvatura e in quella perpendicolare al piano del collider. Questo porta ad un valore della luminosità $L= 10^{34}$ $cm^{-2}s^{-1}$ anche se, per il primo anno LHC, lavorerà ad una luminosità inferiore di un ordine di grandezza. Tale fase verrà sfruttata dall'esperimento LHCb che necessita d'una luminosità minore.
Lungo il percorso vi sono otto stazioni lineari dove avviene l'intersezione tra i due fasci di protoni: quattro sono adibite al monitoraggio dei fasci mentre, nelle rimanenti, verranno situati quattro apparati sperimentali di ATLAS, CMS, ALICE e LHCb (fig. 1.1).

ATLAS [Ref.1A] e CMS [Ref. 1B] sono due grandi spettrometri che coprono buona parte dell'angolo solido progettati per sfruttare al massimo la grande energia nel centro di massa che LHC può generare ($\sqrt{s} = 14TeV$). Sono realizzati, quindi, per lo studio della fisica delle particelle ad alto momento traverso p_t e, in particolare, per la ricerca del bosone di Higgs. Anche se si prefiggono gli stessi obbiettivi, differiscono per le tecnologie impiegate e per i metodi di misura delle traiettorie e degli impulsi delle particelle prodotte nelle collisioni.
L'esperimento ALICE [Ref.1C] studierà, mediante l'esame delle collisioni tra nuclei di piombo, la formazione d'un nuovo stadio della materia, il *quark-gluon plasma,* che si suppone fosse presente nei primi istanti del BigBang.

Infine LHCb sarà interamente dedicato allo studio della violazione del CP nei decadimenti dei mesoni B ($b\bar{b}$).
Nella seguente tabella riportiamo le luminosità richieste da tali esperimenti.

$\sqrt{s}$	14 TeV
circonferenza	26.7 km
$f_{\text{bunch crossing}}$	40 MHz
luminosità di progetto	$10^{34}\text{cm}^{-2}\text{s}^{-1}$
luminosità ATLAS, CMS	$10^{34}\text{cm}^{-2}\text{s}^{-1}$
luminosità LHCb	$2 \times 10^{32}\text{cm}^{-2}\text{s}^{-1}$

Tabella 1.2: Caratteristiche dell'acceleratore LHC.

Si osservi che, come già detto, LHCb necessita d'una luminosità assai minore di quella nominale, che è essenziale per il corretto funzionamento del rivelatore.
Infatti il numero d'interazioni per incrocio N è dato da:

$$N_{LHC} = \frac{\sigma_{anelastica} \cdot L_{LHC}}{f_{bunch\_cross\,ing}} = \frac{(80 \times 10^{-27}\, cm^2)(10^{34}\, cm^{-2} s^{-1})}{40 MHz}$$

$$\approx 20 \qquad \text{interazioni per incrocio}$$

che è un numero troppo elevato sia per il trigger, sia per la ricostruzione dell'evento.
ATLAS e CMS sono immuni da questo problema perché studiano solo gli eventi ad alto momento traverso e quindi eliminano molti degli eventi di fondo che si trovano ad angoli piccoli: LHCb si propone di studiare proprio questi eventi e necessita perciò d'una riduzione della luminosità per ridurre il numero dei dati registrati.

2. L'Esperimento LHCb

2.1 La fisica di LHCb

Lo scopo principale di LHCb è quello di sovra-determinare i parametri dei Triangoli di Unitarietà del Modello Standard tramite misure, di precisione e con alta statistica, della violazione del CP.[Ref. 4]
Allo scopo si presta particolarmente la gran produzione di quark b (bottom o beauty) che si realizzerà in LHC e che darà origine ad un sistema di mesoni B neutri ($\overline{B}_q^0, B_q^0$), costituiti da un quark di tipo b e uno di tipo d o s. Osservando il decadimento di tali mesoni sarà possibile evidenziare la violazione di CP e, per il teorema CPT, la seguente violazione di T, così come nel sistema dei K neutri. Rispetto ai mesoni K, i mesoni B godono però del seguente vantaggio: essendo la massa dei quark b circa 30 volte maggiore di quella dei quark s [Ref.5] (che prendono il posto dei b nei mesoni K) sono possibili previsioni teoriche del Modello Standard estremamente precise, poiché diventa trascurabile il contributo apportato dalle interazioni forti e quindi le correzioni della QCD.
In particolare verranno studiati i canali di decadimento rari del mesone B nei quali si verifica la violazione CP, cosa resa possibile soltanto dall'alta statistica fornita da LHC.

2.1.1 Violazione di CP nei decadimenti dei mesoni B.

I mesoni neutri B vengono prodotti mediante interazione forte ma decadono secondo interazione debole che non conserva il beauty. .[Ref. 4]

Essi sono stati legati di coppie quark-antiquark, contenenti il quark $\bar{b}$ e un altro quark, che danno origine ai quattro mesoni B, due neutri e due carichi, con le rispettive antiparticelle, riportati nella tabella 2.1 insieme alle rispettive masse e vite medie.

mesone	$\bar{q}q$	M(MeV)	$\tau(ps)$
B_u^+	$\bar{b}u$	5279.0 ± 0.5	1.653 ± 0.028
B_d^0	$\bar{b}d$	5279.4 ± 0.5	1.548 ± 0.028
B_s^0	$\bar{b}s$	5369.6 ± 2.4	1.493 ± 0.062
B_c^+	$\bar{b}c$	$(6.4 \pm 0.4) \times 10^3$	$0.46^{+0.18}_{-0.16}$

Tabella 2.1: Il sistema dei mesoni B

Del sistema dei mesoni B, per studiare la violazione di CP, è utile considerare i mesoni neutri B_d^0 e B_s^0.

In analogia allo studio del decadimento dei mesoni K neutri, bisogna considerare due autostati deboli B_L e B_H scritti nel seguente modo:

$$|B_L\rangle = p|B^0\rangle + q|\overline{B^0}\rangle$$

$$|B_H\rangle = p|B^0\rangle - q|\overline{B^0}\rangle$$

dove p e q sono coefficienti complessi che rispettano la condizione di normalizzazione $|p^2|+|q^2|=1$.

A differenza dei mesoni K, però, nel sistema così definito non è possibile osservare la violazione CP cercando direttamente decadimenti vietati: infatti non è possibile creare un fascio puro di B_L o di B_H perché è piccola la differenza fra le rispettive vite medie e quindi fra le rispettive larghezze di decadimento Γ. In particolare si può dimostrare che $\Delta\Gamma/\Gamma \approx 4\times10^{-3}$ per B_d^0 e $\Delta\Gamma/\Gamma \approx 0.1$ per B_s^0.

Per osservare la violazione di CP bisogna comparare i *rate* di decadimento dello stato iniziale di B^0 e di $\overline{B}^0$ oppure misurarne l'andamento temporale.

Si possono distinguere due classi di violazione di CP:

1) Violazione indiretta

Riguarda i decadimenti semileptonici.

Nei decadimenti semileptonici B^0 decade in un leptone positivo e $\overline{B}^0$ in uno negativo.

Il rapporto:

$$A_{sl} = \frac{\dot{N}(\overline{B^0} \to l^- X) - \dot{N}(B^0 \to l^+ X)}{\dot{N}(\overline{B^0} \to l^- X) + \dot{N}(B^0 \to l^+ X)}$$

è detto asimmetria semileptonica. Un'asimmetria non nulla testimonia che una parte dei B decade nel leptone col segno vietato dalla conservazione di CP.

Tale violazione è associata ad un cambiamento di sapore 2 che è previsto essere molto piccolo nei decadimenti d B e la sua misura, resa possibile dalla gran quantità d'eventi disponibili, rappresenta un fertile campo di ricerca di nuova fisica fuori dal Modello Standard.

2) Violazione diretta

Riguarda tutti i canali in cui le ampiezze di decadimento hanno fasi deboli differenti. Infatti mentre l'interazione elettromagnetica e quella adronica possono sempre essere definite reali, per quella debole questo non è possibile.

Se si considera il decadimento dei B^0 e di $\overline{B}^0$ verso lo stesso autostato di CP l'asimmetria di tali canali di decadimento risulta proporzionale al seno della loro differenza di fase:

$$A_{f(t)} = \sin[2(\phi_{mix} + \phi w)]\sin(2\Delta M t)$$

dove $2\phi w$ è la differenza di fase tra il canale di decadimento del B^0 e quello del $\overline{B}^0$, $2\phi_{mix}$ la fase del fenomeno del mixing tra B^0 e $\overline{B}^0$ e ΔM è la differenza di massa tra i due stati B_L e B_H, quantità tutte legate ai triangoli d'unitarietà del Modello Standard.
La misura diretta dell'asimmetria permette di determinare questi parametri da cui si può risalire agli angoli del triangoli d'unitarietà.

Per misurare accuratamente la dipendenza temporale dell'asimmetria è necessario trovare le ampiezze di decadimento in funzione del tempo del mesone B: perciò è necessaria una risoluzione spaziale che permetta di discriminare la distanza tra il vertice primario, in cui il mesone viene creato, e quello secondario, in cui decade. Per esempio, se si considera un B^0 con energia di 200 GeV si trova che:

$$\gamma = \frac{E}{m} = \frac{200GeV}{5GeV} = 40$$

e la particella percorre mediamente una distanza:

$$c\tau\gamma = 464\mu m \times 40 \approx 2mm\,.$$

Questo testimonia che il rivelatore di vertice dovrà avere una risoluzione spaziale eccellente (<<2 mm) per effettuare la misura dell'asimmetria.

Poiché la violazione diretta di CP è legata a cambiamenti di sapore $|\Delta B| = 1$, per misurare l'asimmetria è necessario determinare il sapore del mesone B prodotto (B=±1): il processo d'identificazione del sapore prende il nome di tagging.
In LHCb il tagging viene eseguito sfruttando i leptoni provenienti dal decadimento b→l+q e attraverso i K carichi del decadimento b→c→s.
Nelle interazioni forti vengono prodotte coppie particella- antiparticella B^0 - $\overline{B}^0$:

uno dei due mesoni decade in uno dei canali di interesse per l'esperimento e ne viene completamente ricostruita l'evoluzione, mentre l'altro viene usato per il tagging utilizzando la correlazione tra il segno del sapore e la carica del leptone o del K prodotto:

$$\begin{cases} B^0 \to l^+ & \text{e} \quad B^0 \to K^+ \\ \bar{B}^0 \to l^- & \text{e} \quad \bar{B}^0 \to K^- \end{cases}$$

2.2 Esempi di decadimento dei mesoni B

Nella tabella seguente sono riportati i canali di decadimento del B^0 che, insieme ai corrispondenti canali per $\overline{B}^0$, sono di particolare interesse per lo studio della violazione di CP.

Decadimento del quark	Decadimento del mesone	Parametro misurato
$b \to c + \bar{c}s$	$\mathrm{B_d^0} \to J/\psi\ K_S$	β
$b \to u + \bar{u}d$	$\mathrm{B_d^0} \to \pi^+\pi^-$	$\beta + \gamma$
$b \to c + \bar{c}s$	$\mathrm{B_s^0} \to J/\psi\ \phi$	$\delta\gamma$
$b \to u + \bar{c}s$	$\mathrm{B_d^0} \to D^{*\mp}\pi^{\pm}$	$\gamma - 2\beta$
$b \to c + \bar{u}s$	$\mathrm{B_s^0} \to D_s^{\mp}K^{\pm}$	$\gamma - 2\delta\gamma$
$b \to c + \bar{u}s$	$\mathrm{B_s^0} \to D^0K^{*0}$	γ

Tabella 2.2: Decadimenti noti dei mesoni B e relativi parametri misurabili.

Vediamo i più importanti più in dettaglio:

1) $B_d^0 \to J/\psi\ K_s$

Questo canale di decadimento è interessante per la presenza della J/ψ con doppia segnatura leptonica: le coppie e^+e^- e $\mu^+\mu^-$ prodotte sono ottimi segnali per identificare il canale.

Il rapporto di decadimento vale $\Gamma(B_d^0 \to J/\psi\ K_s)/\Gamma_{TOT} = (8.9 \pm 1.2) \times 10^{-4}$.

Questo canale è anche detto *gold plated* perché vi è un assenza di fattori QCD: ciò permette previsioni teoriche più precise che evidenziano gli effetti della violazione CP.

Nella figura 2.1 sono riportati i diagrammi ad albero e a pinguino del decadimento.[Ref.6].

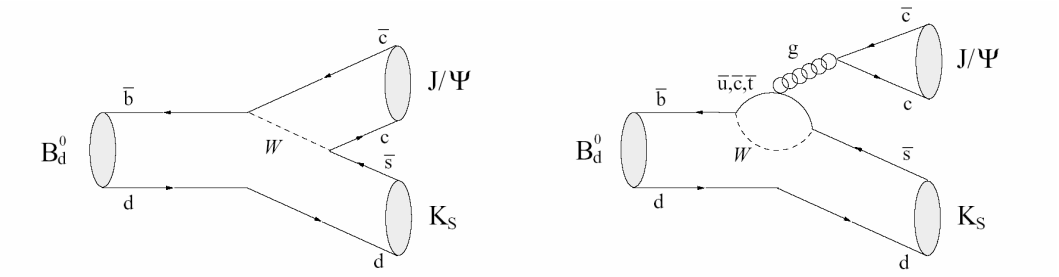


Figura 2.1: Diagrammi ad albero e a pinguino del decadimento $B_d^0 \to J/\psi\ K_s$

2) $B_d^0 \to \pi^+\pi^-$**.**

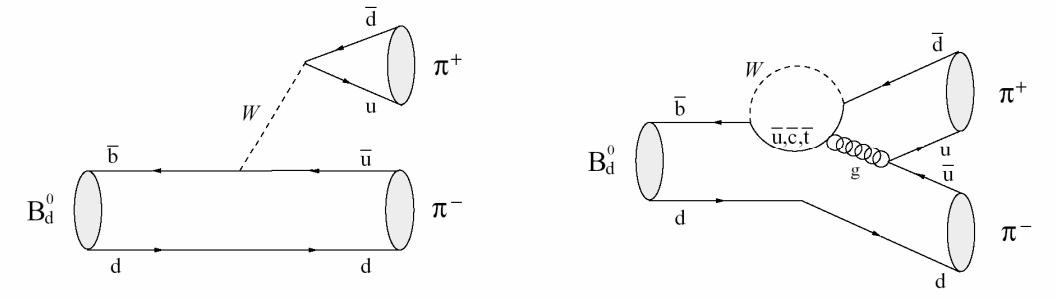


Figura 2.2: Diagrammi ad albero e a pinguino del decadimento $B_d^0 \to \pi^+\pi^-$

Rapporto di decadimento: $\Gamma(B_d^0 \to \pi^+\pi^-)/\Gamma_{TOT} < 1.5\times10^{-5}$

3) $B_s^0 \to J/\psi\ \phi$

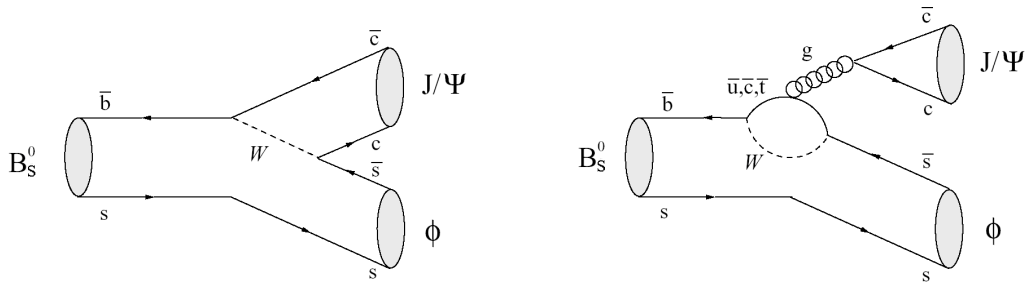


Figura 2.3: Diagrammi ad albero e a pinguino del decadimento $B_s^0 \to J/\psi\ \phi$

Rapporto di decadimento: $\Gamma(B_s^0 \to J/\psi\ \phi)/\Gamma_{TOT} = (9.3 \pm 3.3)\times10^{-4}$

Questo canale è analogo a $B_d^0 \to J/\psi\ K_s$ con l'unica differenza che ora il quark aspettato è un quark s.

Siccome lo SM fornisce valori trascurabili riguardo al contributo alla violazione di CP da parte di questo canale di decadimento esso rappresenta un ottimo strumento per la ricerca d'una nuova fisica.

2.3 Vantaggi della fisica di LHCb

La fisica del b in LHCb gode dei seguenti vantaggi [Ref. 7]:

- La **sezione d'urto relativa** molto alta ($\sigma_{b\bar{b}} \approx 500mb$ $\frac{\sigma_{b\bar{b}}}{\sigma_{anelastica}} \approx 0.01$), che corrisponde a $10^{12}\, b\bar{b}$ /anno fornisce un'alta statistica che permetterà misure precise.

- La **luminosità** appositamente regolata a L = 2x10^{32} cm^{-2}s^{-1} permette di ottenere $\nu = 0.4$ interazioni anelastiche visibili per ogni collisione dei fasci (bunch crossing). La bassa probabilità di avere interazioni multiple permette facilmente di isolare un evento semplificando la ricostruzione degli eventi.

- Alle **alte energie** disponibili b e $\bar{b}$ sono correlati spazialmente, nel senso che sia gli adroni contenenti b, che quelli contenenti $\bar{b}$, sono prodotti in avanti nello stesso cono, come si può vedere chiaramente nel seguente istogramma calcolato dal generatore di eventi PYTHIA.

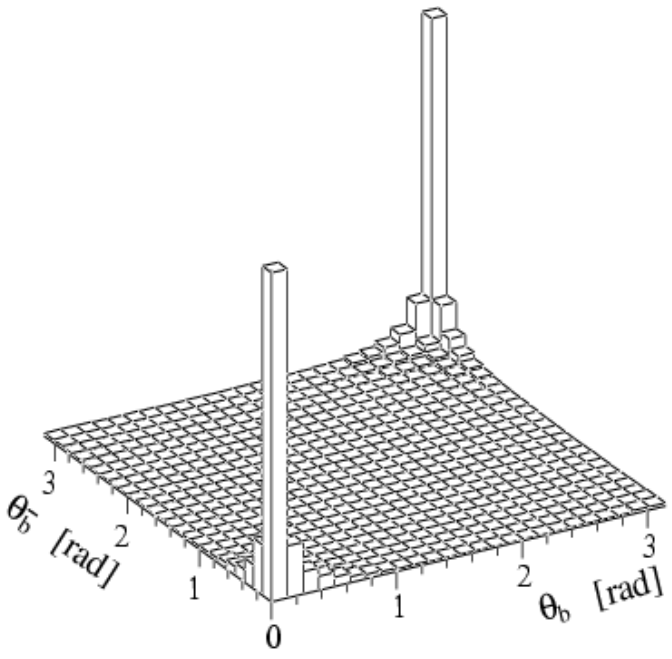


Figura 2.4: Gli angoli polari dei quark b e $\bar{b}$ calcolati dal generatore di eventi PYTHIA. Si vede chiaramente che la distribuzione è fortemente

piccata intorno all'angolo nullo ed è quindi conveniente scegliere, per il rivelatore, una geometria a *braccio singolo*.

- **Alto momento del B ($\beta\gamma \approx 14$)** che vuol dire separazione tra il vertice primario e quello secondario del decadimento del B tale da permetterne facilmente l'identificazione. $\beta\gamma c\tau \approx 7mm$

In virtù di queste proprietà nel paragrafo successivo verrà descritta la struttura del rivelatore di LHCb.

2.4 Struttura del rivelatore

Per il rivelatore di LHCb ha una configurazione a braccio singolo che permette una copertura angolare che va dai 10 ai 300 mrad nel piano xz e da 10 a 250 mrad nel piano yz. La scelta d'una geometria a braccio singolo è giustificata dalla correlazione spaziale, descritta nel paragrafo precedente, che permette di ottenere un'accettanza, sia per il B^0 che per il $\overline{B}^0$, paragonabile a quella d'un rivelatore centrale, che coprirebbe tutto l'angolo solido, ma avrebbe costi di realizzazione più elevati.

In figura 2.5 è riportato la distribuzione dell'impulso dell'impulso del B^0 misurata attraverso i pioni del decadimento $B_d^0 \to \pi^+\pi^-$ in tutto l'angolo solido e quella nel cono d'accettanza del rivelatore: si nota facilmente che più del 30% dei pioni prodotti finiscono nel cono d'accettanza.

Una serie di rivelatori, di diversa tecnologia, disposti lungo la traiettoria delle particelle consentono di effettuare il trigger e il tagging delle particelle. La figura 2.6 mostra una sezione longitudinale e trasversale del rivelatore. [Ref. 8]

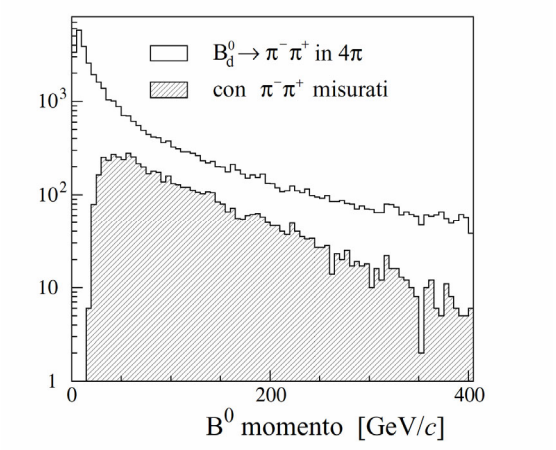


Figura 2.5: Distribuzione dell'impulso per $B_d^0 \to \pi^+\pi^-$ in 4π e quella misurata rivelando sia π^+ che π^- nello spettrometro.

Intorno al punto d'intersezione è situato il rivelatore di vertice VELO. Tale rivelatore ha il compito di contare i vertici primari e discriminare gli eventi indesiderati che contengono più d'un interazione protone-protone (*pile-up veto counter*). Vista la distanza fra tali vertici, necessita di un'alta risoluzione spaziale. Il VELO è composto da 42 piani di silicio di forma semicircolare disposti ad un centimetro dal fascio durante la presa dati e letti mediante strip circolari e radiali (fig.2.7-2.8). Tale rivelatore fornisce una risoluzione spaziale sul vertice primario di 40 μm e una risoluzione spaziale sul tempo proprio di 40 fs. [Ref.9]

Le 11 stazioni T1-T11 costituiscono il sistema principale di tracciamento, insieme ad un potente magnete che sviluppa un campo verticale di 4 Tesla. Le prime stazioni, in particolare le due più vicine al punto di collisione, sono caratterizzate da una risoluzione maggiore.

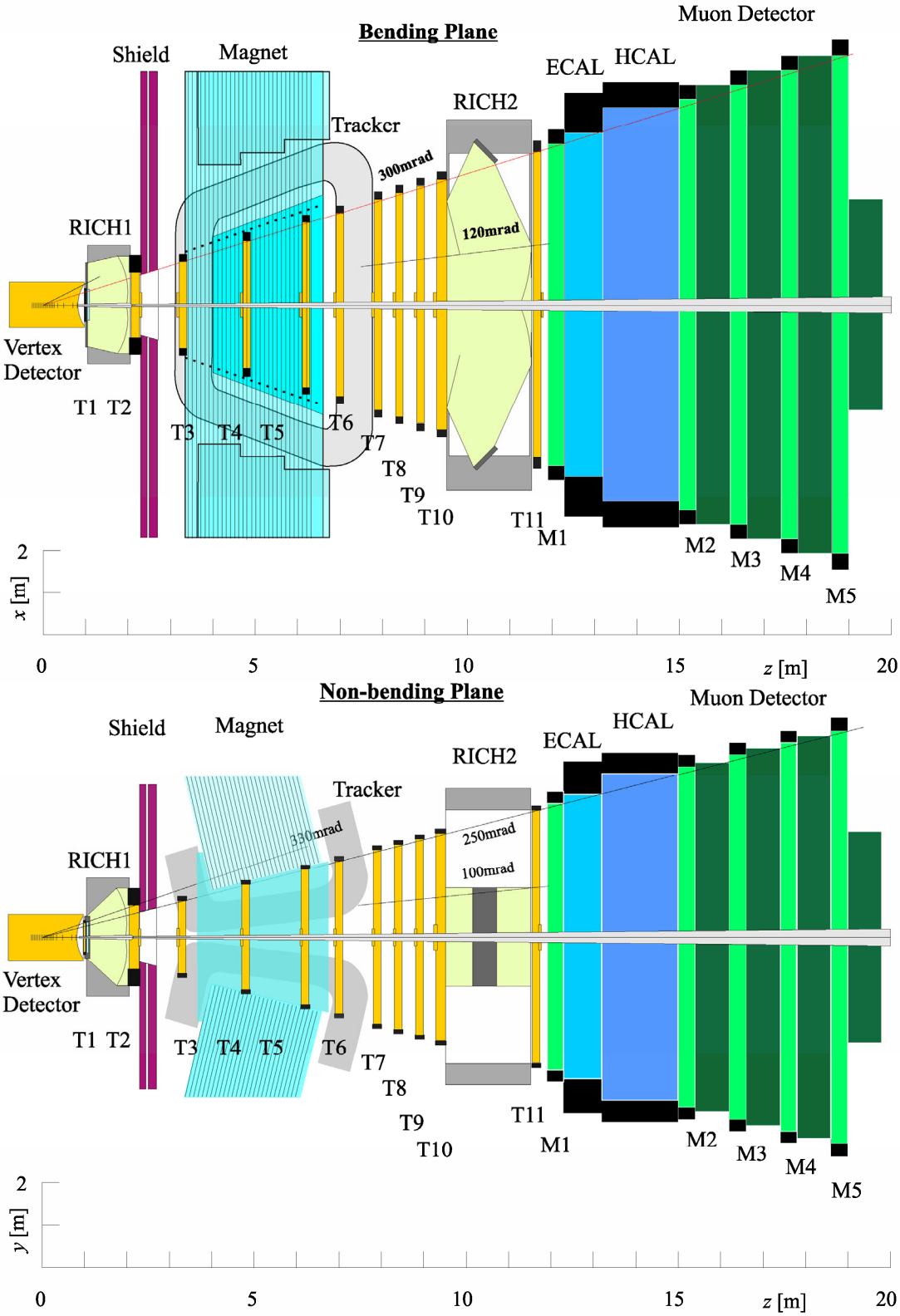


Figura 2.6: Sezione longitudinale e trasversare del rivelatore di LHCb

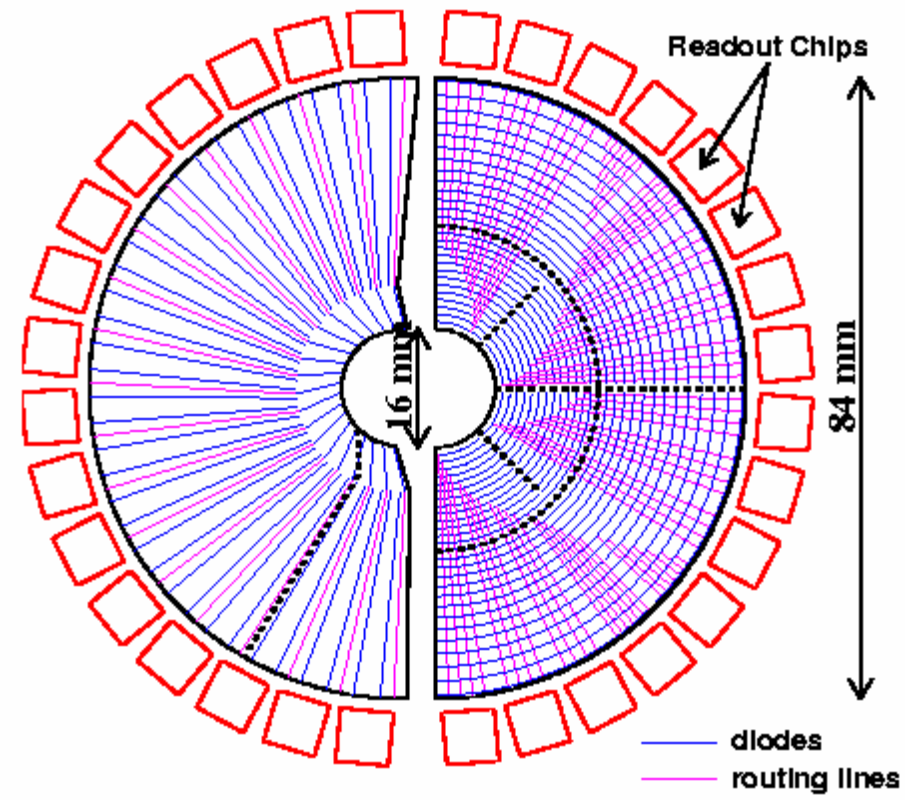


Figura 2.7: Sezione del rivelatore VELO

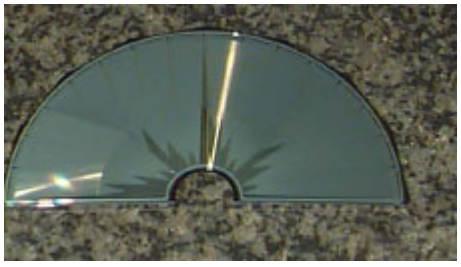


Figura 2.8: sensore al silicio del rivelatore VELO.

Il RICH1 (Ring Imaging Cherenkov) e il RICH2 sono necessari per il riconoscimento delle particelle, soprattutto per distinguere K da π al fine di permettere il riconoscimento del canale di decadimento e il tagging del sapore. Essi coprono tutto l'angolo solido tra 10 e 330 mrad che corrisponde ad un intervallo d'impulso tra 1 GeV e 150 GeV ($1 < p < 70$ GeV/c per il RICH1 e $20 < p < 150$ GeV/c per il RICH2). [Ref.10].

L'ampio range d'impulsi è indispensabile per identificare sia le particelle ad alto momento trasferito, permettendo così di sopprimere il fondo, sia gli adroni con basso momento, necessari per il tagging del sapore.

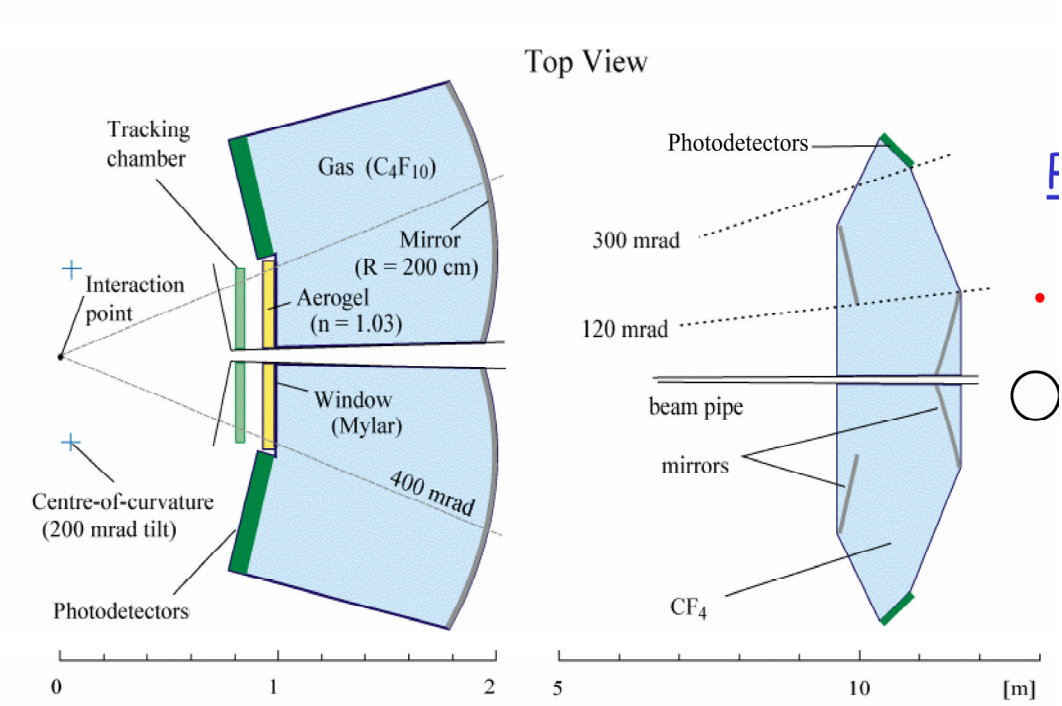


Figura 2.9: Sezione del RICH1 e del RICH2

Il RICH1 è posto subito dopo il rivelatore di vertice ed è costituito da due radiatori, uno ad aereogel di silicio e uno a gas con $C_4F_{10,}$ mentre il RICH2 è posto prima dei calorimetri ed è costituito da un solo radiatore con CF_4;

La parte finale di LHCb è composta dai rivelatori dedicati al trigger.

ECAL e HCAL sono due calorimetri, il primo elettromagnetico ed il secondo adronico.

ECAL è costituito da un rivelatore di preshower, composto da uno strato di piombo al fine di filtrare le particelle cariche con energie più basse, da uno strato di scintillatori, e da un calorimetro con tecnologia Shashilik. La luce viene raccolta tramite fibre a spostamento di lunghezza d'onda WLS (wavelength shifting fibers). Il suo scopo è identificare elettroni, fotoni e π^0 e misurarne posizione e momento.

Lo stesso compito per gli adroni è svolto da HCAL, costituito da scintillatori immersi in una struttura di ferro profonda 1.5 m, anch'essi letti da WLS.

Infine le cinque stazioni M1-M5 costituiscono il rivelatore di muoni, elemento chiave dell'apparato sperimentale, che permette l'identificazione e il trigger di primo livello per i muoni. Questo è indispensabile per l'esperimento: sia perché i muoni sono presenti nello stato finale dei decadimenti dei B neutri con violazione CP, in particolare nei due "gold-plated" $B_d^0 \to J/\psi(\mu^+\mu^-)\,K_s$ e $B_s^0 \to J/\psi(\mu^+\mu^-)\,\phi$, sia perché i muoni provenienti dai decadimenti semi-leptonici permettono il tagging del sapore iniziale dei mesoni.

Le loro caratteristiche verranno approfondite in seguito, vista la loro attinenza al lavoro di tesi.

2.4.1 Il trigger

Il compito del sistema di trigger è quello di selezionare i decadimenti dei mesoni B dal fondi delle collisioni anelastiche.

Come già visto LHCb disporrà di circa 10^5 coppie $b\bar{b}$ al secondo: un numero tale di eventi è impossibile da processare interamente. Diversamente da quanto accaduto finora nella fisica del B, in cui si avevano a disposizione pochi eventi ed era necessario un trigger ad altissima efficienza, il trigger di LHCb seleziona soltanto gli eventi con alto momento trasverso.

Il sistema di trigger è diviso in quattro livelli, due hardware e due software:

1. **livello 0 (hardware):** il rate d'ingresso di questo livello è 40 MHz (vale a dire la frequenza di bunch crossing), mentre quello di uscita è di 1 MHz. Esso usa le informazioni provenienti dal calorimetro e dalle stazioni M1-M5 per selezionare gli eventi con adroni, leptoni o fotoni con alto momento trasverso. In funzione del tipo di particella varia il valore della soglia di accettazione nell'intervallo tra 1 GeV e 3.5 GeV. Inoltre il *pile-up veto* rigetta tutti gli eventi caratterizzati da interazioni multiple per singolo bunch crossing.

2. **livello 1 (hardware):** il rate d'ingresso è di 1 MHz mentre quello di uscita di 40 kHz. Il livello 1 usa le informazioni raccolte dal VELO e dalle prime stazioni di tracking per identificare i vertici secondari dislocati vicino ai vertici primari, caratteristico dei decadimenti di mesoni B.

3. **livello 2 (software):** ha un rate d'ingresso di 40 kHz e uno d'uscita di 5 kHz. Questo livello accede alle informazioni dell'intero evento

aggiungendo l'informazione sull'impulso ricavate dal tracker alle tracce dei vertici secondari.

4. **livello 3 (software):** ha un rate d'ingresso di 5 kHz e uno d'uscita di 200 Hz. Combinando le informazioni di tutti i rivelatori e ricostruisce lo stato finale facendo uso di algoritmi d'analisi.

Infine le informazioni vengono salvate su un nastro alla frequenza di 200 Hz.

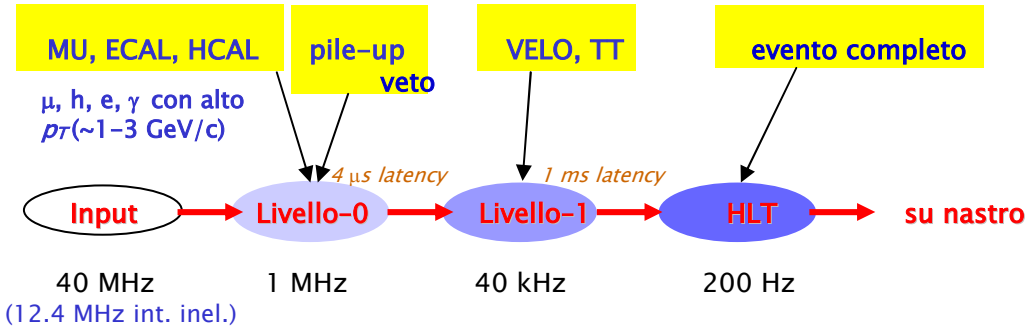


Figura 2.10: diagramma di flusso dei quattro livelli di trigger di LHCb.

2.4.2 Il tagging

È già stato sottolineato che per misurare le asimmetrie di CP bisogna conoscere il sapore del B che si sta osservando: questa operazione è detta tagging.
Siccome l'interazione forte porta sempre alla creazione si coppie particella-antiparticella B^0-$\overline{B}^0$ il secondo B viene utilizzato per il tagging.

A seconda del metodo con cui viene effettuato il tagging può essere *leptonico* o *kaonico*.

Si parla di *tagging leptonico* quando si esamina il leptone proveniente da un decadimento semi-leptonico del tipo $B^0 \to l^+ X$ .Questo metodo fornisce un numero piccolo di errori, ma ha una scarsa efficienza, dovuta alla piccola frazione di decadimento (circa il 10% degli eventi).

Le principali cause d'errore nel tagging leptonico sono dovute al mixing dei B^0 e dei leptoni dal decadimento $b \to c \to l$ che però possono essere tagliati con facilità perché caratterizzati da un basso momento trasverso p_t. Il grafico in figura 2.11 riporta il momento trasverso dei muoni provenienti direttamente dal b e quello dei muoni provenienti dal c per un decadimento $B_d^0 \to \pi^+ \pi^-$. Si vede chiaramente che un taglio a 1.5 GeV permette di ottenere una buona separazione.

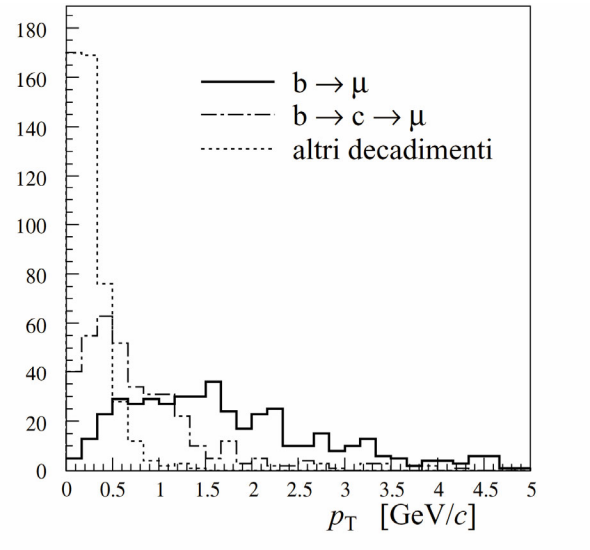


Figura 2.11: Distribuzione del p_t, per muoni dai decadimenti b→μ, b→ c→ μ e da altre fonti, in eventi in cui il secondo B decade in $\mu^+\mu^-$.

Si parla invece di *tagging kaonico* se si usa il kaone prodotto nei decadimenti del tipo $B^0 \to K^+ X$ per l'identificazione del segno del sapore. Questa volta l'efficienza è elevata (78% degli eventi), ma bisogna riuscire a separare i $K^\pm$ dal

fondo dei pioni carichi. Per questo scopo gioca un'importanza fondamentale il RICH.

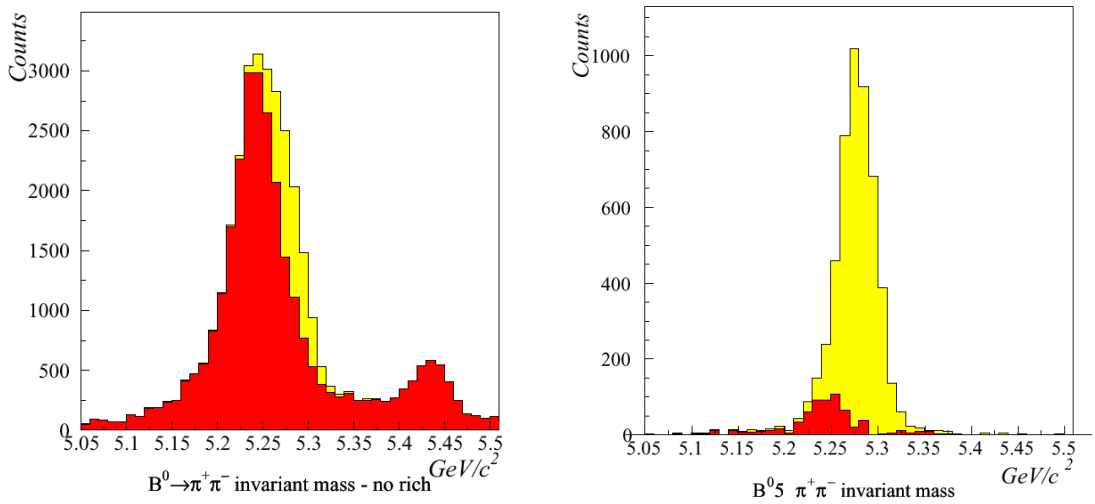


Figura 2.12: spettro della massa invariante con e senza il RICH. Si può vedere chiaramente la rimozione del fondo (in rosso) operata dai RICH.

Nella figura 2.13 si può vedere che i K provenienti dal decadimento del b hanno un grande parametro d'impatto d_0. Eseguendo un taglio richiedendo che $\frac{d_0}{\sigma_d} > 3$, dove σ_d è l'errore su d_0, si ottiene anche stavolta una buona separazione dal fondo.

Per entrambi i tagging vale la regola che il segno della carica del leptone (e o μ) o del K è opposto al segno del sapore: cioè i B^0 (B=-1) decadono con l^+ e K^+ e i $\overline{B}^0$ (B=+1) in l^- e K^-.

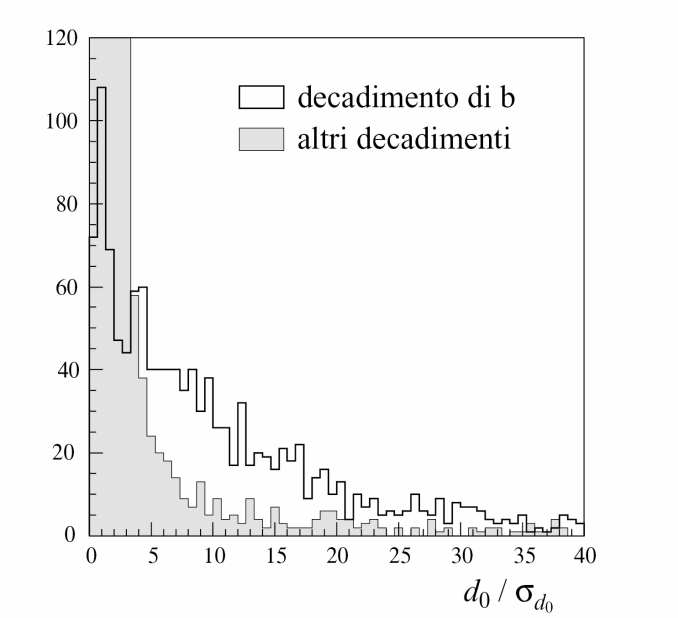


Figura 2.13: Distribuzione del parametro di impatto per caoni con p_t>0,8 GeV dal decadimento del b e da altre fonti, in eventi in cui uno dei due B decade in $\mu^+\mu^-$.

3. Il rivelatore di muoni

3.1 Il rivelatore di muoni

Il rivelatore di muoni, composto essenzialmente dalle 5 camere M1-M5, assolve un compito fondamentale nell'esperimento permettendo il trigger di livello 0 e l'identificazione dei muoni. Il trigger muonico si basa sulla ricostruzione della traccia (tracking) e sulla misura del momento trasverso delle particelle con una risoluzione del 20%.

Perché un evento sia preso in considerazione dovrà interessare tutte le stazioni M1-M5 che sono messe in AND: questo assicura la presenza del muone come particella con alto potere penetrativo. Le cinque stazioni hanno caratteristiche diverse dovute all'attenuazione subita dal fascio man mano che ci si allontana dal punto d'intersezione: la prima stazione si trova a 12.1 m, fra il tracker T11 e il calorimetro elettromagnetico, e subisce lo schermo del RICH2, mentre le altre rispettivamente a 15.2 m (M2), 16.4 m (M3), 17,6 m (M4) e 18.8 m (M5) e sono schermate da filtri di ferro. Il fascio di particelle viene quindi notevolmente attenuato, per quanto riguarda il rate e l'energia delle particelle che lo compongono, e richiede caratteristiche diverse per le cinque stazioni. Le stesse considerazioni si possono fare se ci si muove trasversalmente dal centro del fascio. Inoltre tale apparato funge da filtro del fondo di particelle a bassa energia fornendo una soglia di 5 GeV per l'energia del muone, necessaria per penetrare tutto il materiale ed arrivare fino alla stazione M5.

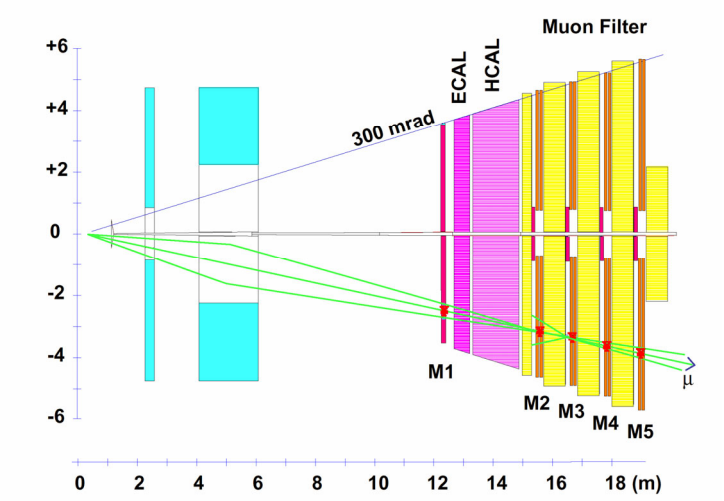


Figura 3.1: le componenti del rivelatore di muoni.

Le 5 stazioni di rivelazione hanno una struttura a pad bidimensionali. Al fine di ottimizzare il numero delle pads e conseguentemente quello dei canali logici del rivelatore, le dimensioni delle pads logiche segue la geometria proiettiva del rivelatore. Ogni stazione è divisa in quattro regioni (R1-R4) caratterizzate da diverse dimensioni delle pads. Più in dettaglio la superficie delle pad aumenta d'un fattore due passando dalla regione centrale, più vicina al fascio di particelle, a quella più esterna: così viene mantenuto pressoché costante il rapporto tra la superficie d'ogni pad e il rate di particelle aspettato. Per lo stesso motivo le dimensioni delle pads aumentano da una stazione all'altra.

Diverso è anche il rapporto y/x fra la lunghezza trasversale e quella longitudinale: la dimensione lungo x, dove agisce l'effetto di curvatura del campo magnetico, è determinata dalla precisione richiesta per la misura di p_t mentre, quelle in y, dalla richiesta di rigettare il fondo, poiché le particelle con basso p_t vengono deviate ad angoli più grandi ed escono rapidamente dall'angolo solido di interesse per l'esperimento, permettendo di usare per y una risoluzione inferiore. È stato scelto un rapporto y/x pari a 2.5 nella stazione M1, 5 in M2 e M3 e 1,25 in M4-M5 che non contribuiscono alla misura di p_t ma solo a confermare la presenza di muoni penetranti.

	M1	M2	M3	M4	M5
R1	1 × 2.5	0.5 × 2.5	0.5 × 2.5	2 × 2.5	2 × 2.5
R2	2 × 5	1 × 5	1 × 5	4 × 5	4 × 5
R3	4 × 10	2 × 10	2 × 10	8 × 10	8 × 10
R4	8 × 20	4 × 20	4 × 20	16 × 20	16 × 20

Tabella 3.1: Dimensioni (xy, in cm^2) delle pad logiche nelle diverse zone delle cinque stazioni per i muoni.

Una stima dei diversi rate di particelle, a cui sono soggette le diverse regioni nelle cinque stazioni, è stata effettuata tramite l'uso di due programmi di simulazione, GCALOR e MARS [Ref 11, 12, 13]. Visti i risultati discordi dei due simulatori, MARS ha predetto un rate doppio di particelle rispetto a GCALOR, è stato fissato un fattore di sicurezza cinque volte superiore a quello previsto da GCALOR per le stazioni M2-M5 e due volte superiore per la stazione M1, che è posta davanti ai calorimetri e quindi meno affetta da incertezza. I risultati sono riportati nella tabella 3.2 per una luminosità $L = 5 \times 10^{32}\ cm^{-2} s^{-1}$.

	M1	M2	M3	M4	M5
R1	$230 \cdot 10^3$ $460 \cdot 10^3$	$7.5 \cdot 10^3$ $37.5 \cdot 10^3$	$2 \cdot 10^3$ $10 \cdot 10^3$	$1.3 \cdot 10^3$ $6.5 \cdot 10^3$	880 $4.4 \cdot 10^3$
R2	$93 \cdot 10^3$ $186 \cdot 10^3$	$5.3 \cdot 10^3$ $26.5 \cdot 10^3$	650 $3.3 \cdot 10^3$	430 $2.2 \cdot 10^3$	350 $1.8 \cdot 10^3$
R3	$40 \cdot 10^3$ $80 \cdot 10^3$	$1.3 \cdot 10^3$ $6.5 \cdot 10^3$	$200 \cdot 10^3$ $1.0 \cdot 10^3$	150 750	130 650
R4	$12.5 \cdot 10^3$ $25 \cdot 10^3$	230 $1.2 \cdot 10^3$	83 415	50 250	45 225

Tabella 3.2: Rate di particelle in kHz/cm^2 nelle varie zone del rivelatore di muoni. Nella prima riga sono riportati i valori calcolati alla luminosità $L=5x10^{32}cm^{-2}s^{-1}$ mentre nella seconda lo stesso valore includendo il fattore di sicurezza.

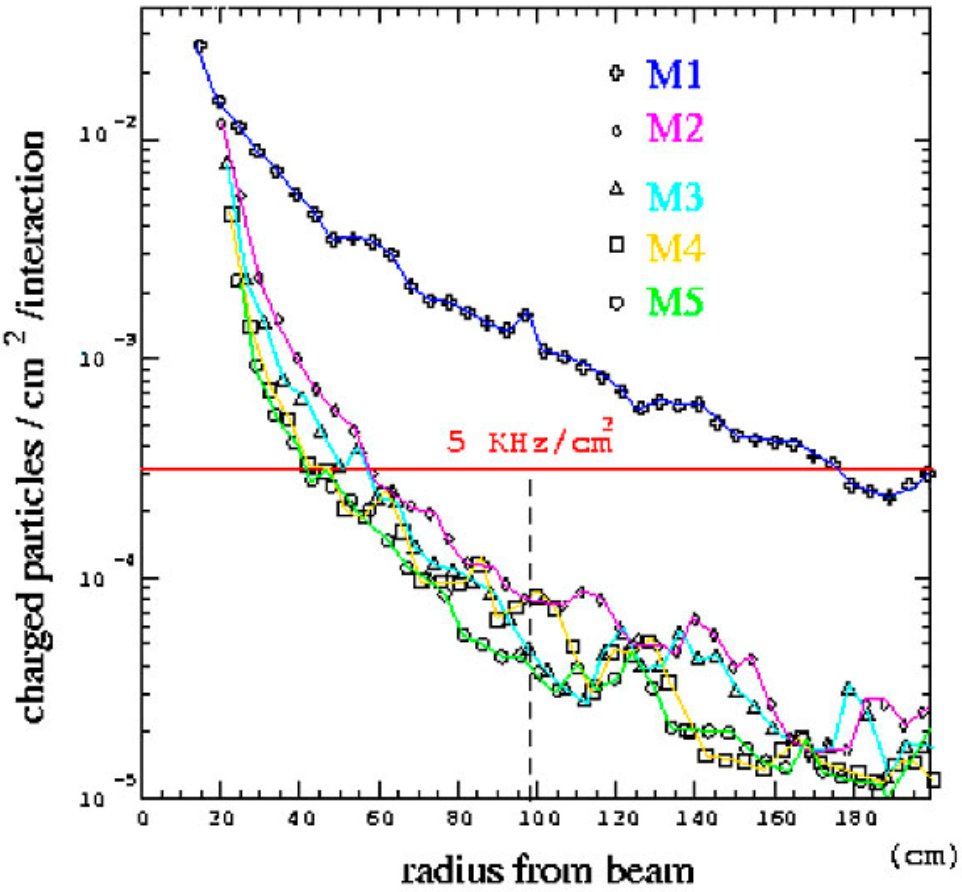


Figura 3.2: Grafico del flusso di particelle in funzione della distanza dal centro del fascio simulato con MARS. Il flusso di particelle più elevato (sopra la retta rossa dei 5 kHz/cm^2) si ha nella stazione M1 (in Blu) e nelle regioni interne di M2-M5. Tanto più si va verso l'esterno, tanto più il flusso viene attenuato, giustificando l'utilizzo d'una geometria meno performante.

La lettura del rivelatore di muoni avviene tramite i segnali in uscita dall'elettronica di front-end che amplifica, forma e discrimina i segnali provenienti dal rivelatore fornendo in uscita un'informazione digitale (canali fisici). Siccome il trigger di livello 0 necessita d'una segmentazione più grossolana di quella fornita a questo stadio, canali fisici provenienti da camere diverse vengono raggruppati in OR logici tramite delle schede intermedie, dette Intermediate Board (IB), poste sul lato del rivelatore di muoni, in modo dipendente dalla loro posizione geografica. Queste permettono la riduzione dei

152832 canali fisici in 25920 logici, che costituiscono gli ingressi per il trigger e per l'acquisizione. [Ref 14].

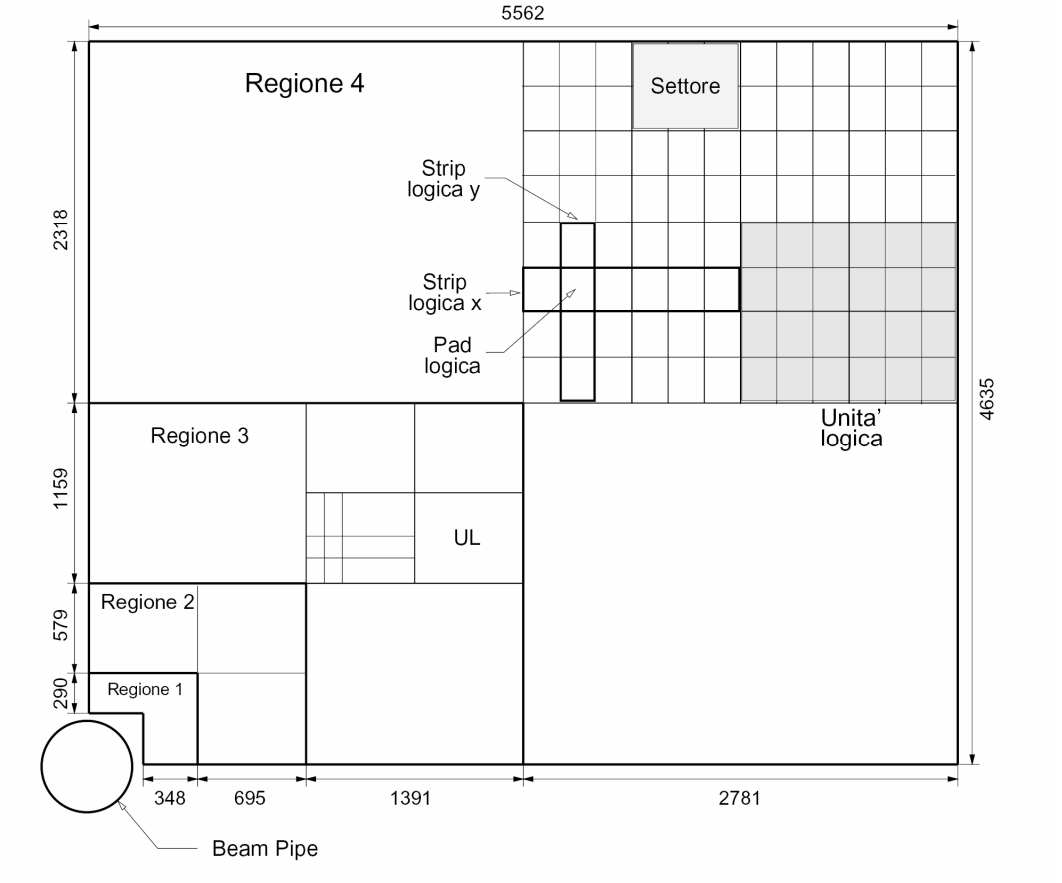


Figura 3.3: Vista frontale d'un quadrante della stazione M4. Le dimensioni delle altre stazioni cambiano secondo la geometria proiettiva del rivelatore. Si noti la suddivisione in settori e in pad e strip logiche.

I canali logici possono rappresentare sia delle pads logiche che delle strips.[Ref.15] Nel primo caso il canale logico identifica una zona ben precisa del rivelatore mentre, nel secondo, l'elettronica di trigger identificherà la pad logica corrispondente dall'incrocio delle strips in x e di quelle in y.

L'uso delle strips permette di diminuire notevolmente il numero dei canali necessari, ma è soggetto all'errore dell'hit fantasma: il passaggio contemporaneo di due particelle accende quattro strip, due orizzontali e due verticali,

identificando quattro pad colpite, due “vere” e due “fantasma” impossibili da distinguere. Il fenomeno è rappresentato nella figura 3.4 .

.

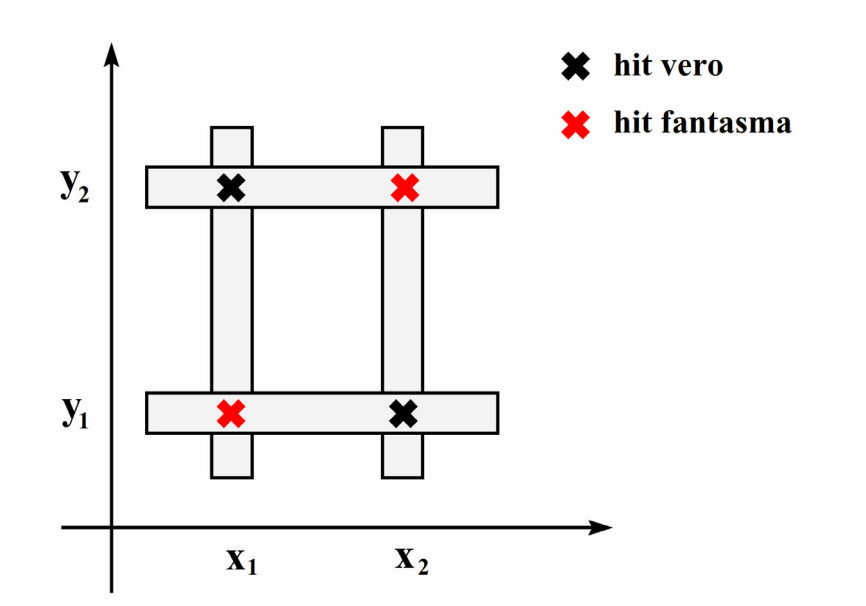


Figura 3.4: Hit fantasma

È evidente che il verificarsi di quest’evento è tanto più probabile quanto più grande è il rate delle particelle, mentre a rate bassi può diventare trascurabile.
Tale fenomeno rende impossibile l’utilizzo di strips nella stazione M1 che è mappata con sole pads. I risultati ottenuti in simulazione hanno permesso invece l’utilizzo di strip di lunghezza opportuna nelle altre stazioni. Nella tabella 3.3 è riportata la divisione dei canali logici in strips e pads.

	M1	M2	M3	M4	M5
R. I	576 pads	336 strips	336 strips	288 pads	288 pads
R. II	576 pads	384 strips	384 strips	168 strips	168 strips
R. I	576 pads	336 strips	336 strips	120 strips	120 strips
R. II	576 pads	336 strips	336 strips	120 strips	120 strips
Totale	6480				
×4	25920				

Tabella 3.3: Numero di pads e strip logiche utilizzate nella varie regioni delle cinque stazioni per i muoni.

Si noti che anche nella regione R.I delle stazioni M4 e M5 non vengono utilizzate strips, poiché, a causa della granularità richiesta, non si otterrebbe una riduzione significativa dei canali logici.

3.1.1 Tecnologia del sistema di rivelazione per i muoni.

La scelta dei rivelatori da usare per le camere M1-M5 è stata condizionata sia dalla tipologia del fondo che dalle finalità dell'esperimento.
Le principali richieste per le camere sono le seguenti:

1. *Alto rate:* cioè basso tempo morto visto l'elevato rate di particelle aspettato. Vedremo che tale richiesta comporterà la scelta di diverse tecnologie per i rivelatori, visto l'incremento del flusso d'un fattore $2x10^3$ fra la regione più esterna della stazione M5 e quella più interna di M1.

2. *Resistenza all'invecchiamento:* la scelta dei materiali dovrà essere tale da assicurare il funzionamento dei rivelatori almeno per i 10 anni in cui si prevede l'utilizzo del rivelatore.

3. *Risoluzione temporale adatta al periodo di BX:* ai fini del trigger muonico è essenziale che ad ogni particella sia assegnato il BX-identifier in maniera non ambigua, cioè che venga rilevato il suo passaggio nel rivelatore in una finestra temporale non più grande del periodo di bunch crossing. In particolare si richiede che questo avvenga con un'efficienza del 95% in una finestra temporale di 20 ns.

4. *Risoluzione spaziale:* si richiede che la risoluzione spaziale sia tale da permettere di determinare il momento trasverso della particella

rivelata con un'incertezza inferiore al 20%. Tale richiesta riguarda in particolare le stazioni M1 e M2. Per soddisfarla bisogna tenere conto anche del cluster-size geometrico del rivelatore scelto.

A causa del diverso rate di particelle a cui sono soggette le varie zone delle cinque camere per i muoni, sono state usate tecnologie diverse per la loro realizzazione.
In tutte le regioni, tranne che nelle ragioni R1 e R2 della camera M1, sono state scelti rivelatori MWPC (Multi Wire Proportional Chambers) mentre nelle rimanenti, caratterizzate da rate superiori a 100 kHz/cm^{-2}, è ancora in fase di studio la scelta della tecnologia, che sembra vertere su dei rivelatori GEM (Gas Electron Multiplier).
Le MWPC di LHCb sono composte da quattro gap di gas messe in OR per ottenere un'ottimizzazione della risoluzione temporale che risulta essere migliore di 3ns, e quindi in ottimo accordo con la richiesta di avere una risoluzione adatta a discriminare bunch crossing diversi. Ogni camera ha una geometria a due gap messe in OR logico, che corrispondono a due canali fisici per l'elettronica di front-end. Tale ridondanza assicura un'efficienza del 99% in una finestra di 20 ns. Ogni strato è composto d'un reticolo di fili spaziati fra loro di 2 mm e una gap di gas, fra i fili anodo e i catodi pad, di 5 mm . Come gas è stata scelta una miscela di Ar, CO_2 e CF_4 nel rapporto 40/50/10 che permette una rapida propagazione delle cariche al suo interno e concorre a soddisfare la richiesta d'una buona risoluzione temporale. [Ref.16]. In tabella 3.4 sono riportate le caratteristiche principali dei rivelatori MWPC.

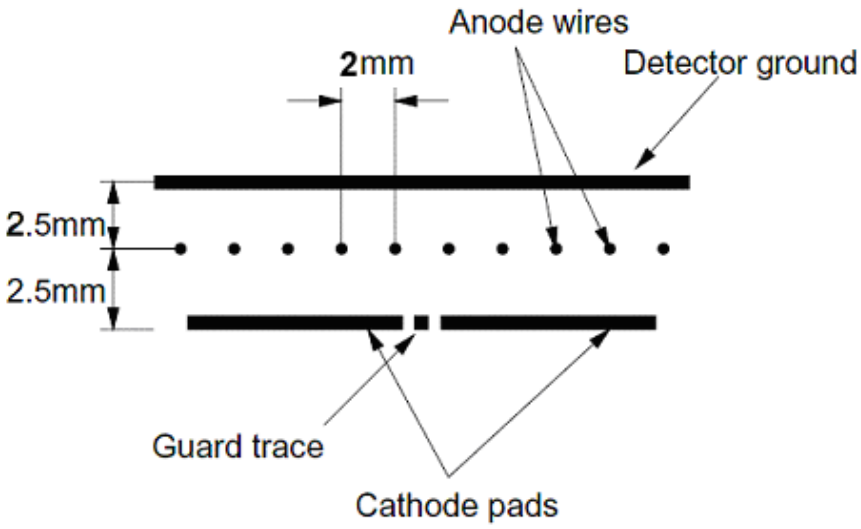


Figura 3.5: schema d'una gap delle MWPC.

Le MWPC sono soggette ad un deterioramento della risoluzione spaziale dovuta al fatto che, la traccia del passaggio d'una particella nel rivelatore, può accendere più d'una pad logica (fig.3.6). Infatti ogni camera è costituita da due strati di pad e quindi la particella può, sotto un certo angolo, entrare in una pad e uscire da un'altra, provocando un hit in entrambe: questo evento è facilmente riconoscibile e non costituisce un problema per quanto riguarda il conteggio della particella, ma provoca un aumento dell'incertezza in termini spaziali.

Questo effetto prende il nome di cross-talk. Una sua stima viene data considerando il numero medio di pad accese al passaggio d'una particella, detto *cluster-size geometrico,* e dipende dall'angolo medio delle tracce, dalla grandezza delle pad e dalla distanza fra gli strati del rivelatore: più tale quantità è maggiore di 1, più la risoluzione spaziale ne risulta intaccata. Nelle camere di LHCb si è deciso di mantenere tale valore al disotto di 1,2 per mantenere alta l'accettanza di M1.

Tale necessità, insieme al grande incremento del rate di particelle nelle regioni più interne di M1, giustificano la scelta d'un tipo di rivelatore diverso in queste zone.

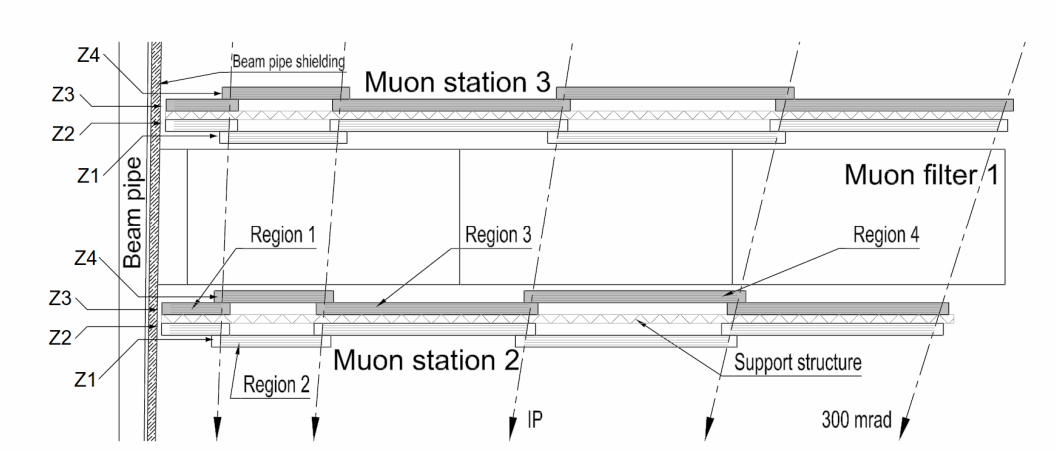


Figura 3.6: Vista parziale del sistema di muoni nel piano y=0. Le frecce indicano il punto d'intersezione. Si noti che, in alcuni casi, la traccia attraversa più d'una pad.

Parametri	Valori di progetto
Gap di gas	5 mm
Spaziatura dei fili	1.5 mm
Diametro dei fili	30 μm
Tensione di lavoro	3.0÷3.2 kV
No. di gap	4
Miscela di gas	Ar / CO_2 / CF_4 (40:50:10)
Ionizzazione primaria	$\simeq$ 100 e^-/cm
Guadagno del gas	$\simeq 10^5$
Soglia	$\simeq$ 3 fC
Carica su 5 mm di traccia	$\simeq$ 0.8 pC

Tabella 3.4: Parametri principali del rivelatore MWPC.

Nelle regioni R1 e R2 di M1 si pensa di utilizzare un rivelatore composto di tre strati di GEM. [Ref.17]

Un rivelatore GEM (Gas Electron Multiplier) è costituito da un foglio di kapton, un polimero con una bassa costante dielettrica quindi molto isolante, dello

spessore di 50 μm, rivestito su entrambi i lati da un sottile strato di rame dello spessore di 5 μm e perforato in superficie con un'alta densità di canali (fig 3.6). Tra i due strati di rame è applicata una differenza di potenziale di 500 Volt producendo così un campo elettrico di 100 kV/cm all'interno dei canali. Tale struttura è posta in una gap di gas fra un catodo e le pad anodo. Gli elettroni creati dal passaggio di particelle ionizzanti nel gas, pilotati dal campo elettrico, vengono diretti verso i GEM dove l'alto campo all'interno dei canali provoca un effetto a valanga moltiplicando il numero degli elettroni nel canale. Infine, gli elettroni secondari prodotti da quest'effetto, vengono rivelati dalle pad sottostanti. Tale struttura permette un guadagno nell'ordine di 10 ÷ 100.

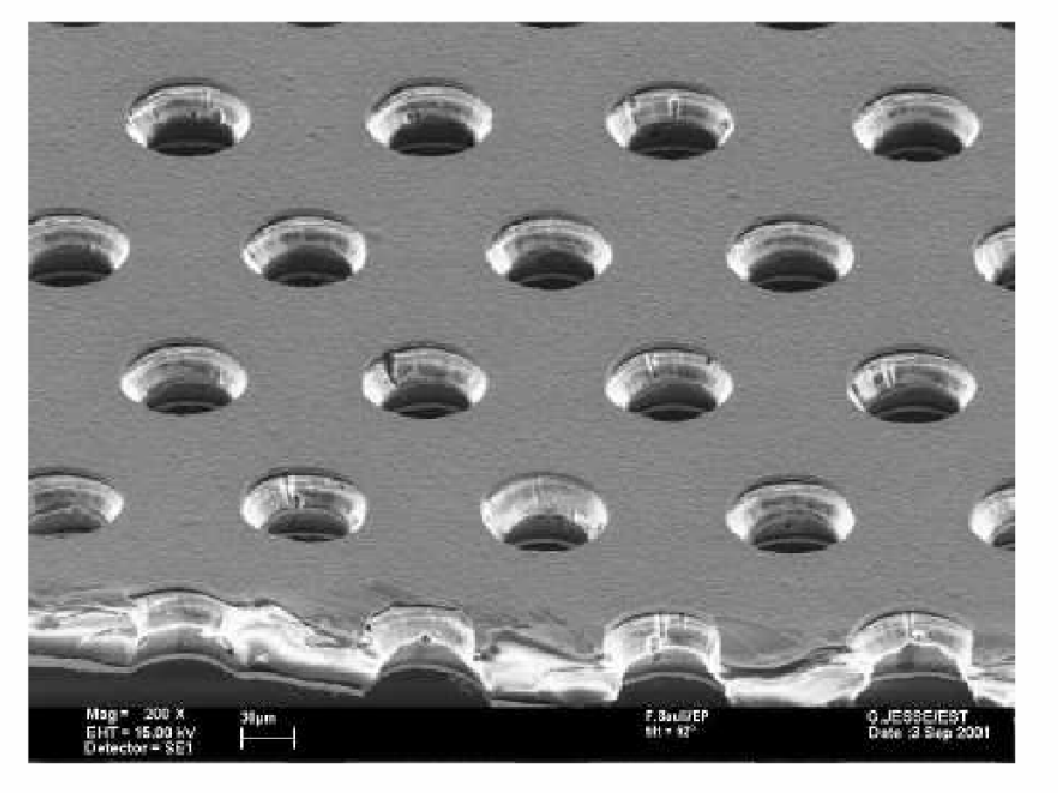


Figura 3.6: Superficie d'uno strato dei rivelatori GEM.

L'utilizzo d'un triplo strato di GEM e d'una miscela di gas composta di Ar, CO_2 e CF_4 nel rapporto 60/20/20 ha permesso di ottenere un'efficienza del 90%

considerando una finestra temporale di 20 ns e quindi in buon accordo con le caratteristiche richieste in LHCb.

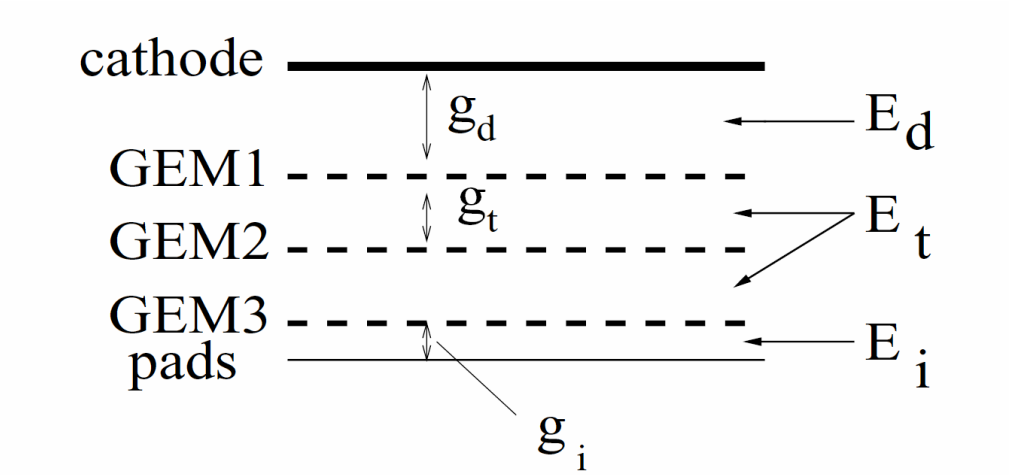


Figura 3.7: Schema delle gap d'un rivelatore GEM a tre strati con catodi pads.

3.1.2 Come avviene il trigger di livello zero

L'identificazione d'un muone avviene cercando gli hit corrispondenti nelle cinque stazioni. In particolare le prime tre necessitano d'una risoluzione maggiore perché, dagli hit di tali stazioni, si ricava la traiettoria della particella rispetto al centro del fascio. Dall'angolo che questa forma con la congiungente al punto d'intersezione si può risalire al momento trasverso del muone.

Il trigger di livello 0 seleziona muoni con p_t alto. Questo livello tratta in maniera indipendente i quattro quadranti cercando le tracce in ciascuno di essi mediante 12 unità di processo raggruppate in gruppi da quattro, nelle regioni R1, R3 e R4, e in gruppi da due in R2 come riportato in figura 3.8.

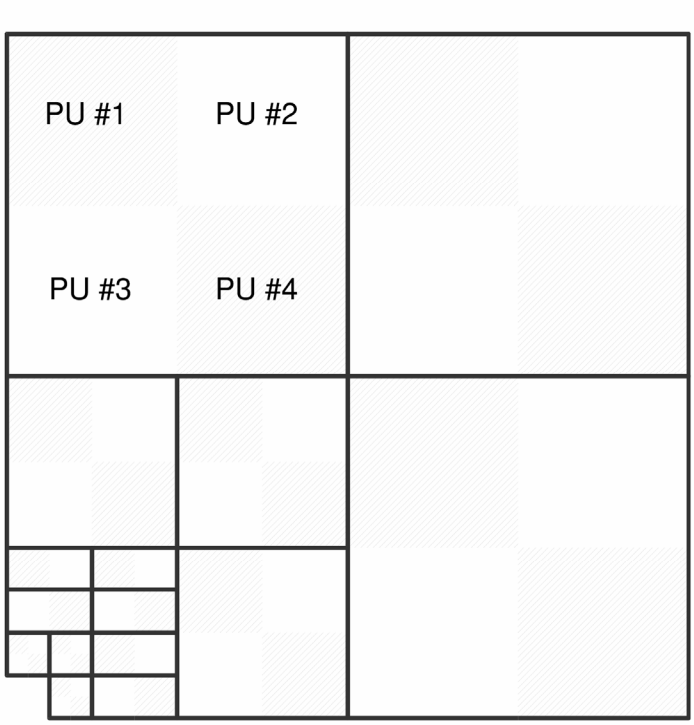


Figura 3.8: Divisione d'un quadrante in unità di processo.

Il processo di trigger avviene nel modo seguente: quando una particella colpisce una delle pad logiche della stazione M3 viene estrapolata la retta passante tra il punto d'intersezione e la pad colpita. Intorno a questa retta viene aperta una finestra nelle stazioni M2, M4 e M5, detta campo d'interesse (FOI), nella quale cercare le altre pad colpite (fig.3.9). Se, all'interno di tale finestra, almeno una pad per ogni stazione è stata colpita la traccia corrispondente è registrata e viene identificata la pad della stazione M2, fra quelle colpite, più vicina alla retta estrapolata. A questo punto si estrapola una nuova retta passante tra questa pad e quella colpita nella stazione M3, in modo da restringere ulteriormente la finestra di ricerca, e s'identifica la pad colpita della stazione M1 più vicina alla nuova retta.

Infine, utilizzando i dati relativi alla pad colpita nelle solo stazioni M1 e M2 viene ricostruita la traccia definitiva del muone.

Considerando la curvatura che la traiettoria subisce nel piano xz a causa del campo magnetico viene calcolato il momento trasverso p_t. Per ogni unità di

processo viene eseguita la misura di p_t per un massimo di due tracce e, per ogni quadrante, sono selezionate le due tracce con momento trasverso maggiore. L'informazione così ottenuta viene, infine, inviata all'unità decisionale.

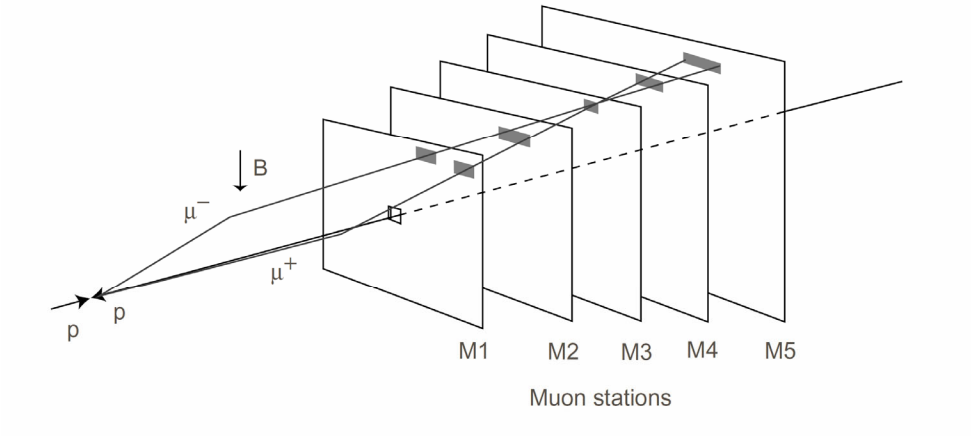


Figura 3.9: Ricostruzione della traccia di due muoni che colpiscono la stazione M3 nello stesso punto; sono evidenziate in grigio le FOI.

4. L'elettronica del rivelatore di muoni.

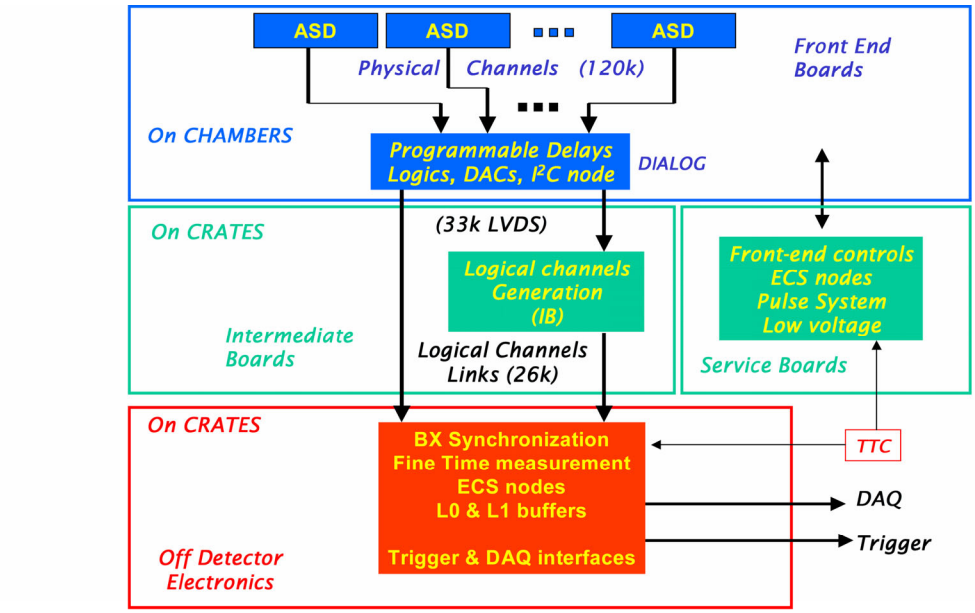


Figura 4.1: Schema dell'elettronica del rivelatore. Possiamo distinguere tre zone in base alla posizione dell'elettronica: una facente parte delle camere stesse (blu), una esterna al rivelatore di LHC ma a diretto con le camere (verde) e una a qualche metro dal rivelatore.

In figura 4.1 è riportata la struttura dell'elettronica che serve il rivelatore di muoni.

Possiamo distinguere tre zone per l'elettronica, in base alla posizione che occupa rispetto al rilevatore, ognuna rispondente a diverse esigenze per quanto riguardo la tolleranza alle radiazioni.

La prima (rettangolo blu) è sul rivelatore e comprende le schede dette di Front-End (FE). L'elettronica di questa zona è soggetta ad un notevole assorbimento di radiazioni, a causa della vicinanza la fascio, che arriva fino ad 1 Mrad/anno.

La seconda zona, in verde, è esterna al rilevatore, ma si trova fissata su di esso e comprende le IB (intermediate boards) e le Service Boards.
L'ultima, evidenziata in rosso, situata qualche metro fuori del rilevatore, contiene le schede ODE (off Detector Electronics). [Ref.18]

La funzione principale dell'elettronica del rivelatore è quella di preparare l'informazione proveniente dalle stazioni M1-M5 per il trigger L0 e L1. Questo corrisponde ad organizzare i 156000 canali fisici in 26000 logici e marcare ogni segnale ricevuto con il proprio "BX identifier", un numero ad otto bit che lo associa al bunch crossing di cui fa parte.
È di fondamentale importanza che il primo passo di questo processo, vale a dire la riduzione dei canali fisici in logici, sia realizzato nello stadio più vicino al rivelatore, in modo da ottenere subito una riduzione di canali riducendo notevolmente la complessità e i costi dell'apparato. Per fare un esempio, se i canali fisici non fossero messi in or nelle camere, sarebbero necessarie 888 Intermediate Boards (schede che ricevono parte dei canali fisici, cap. 4.2) nello stadio successivo, invece che 168. Questa scelta comporta in ogni caso due svantaggi non trascurabili. In primo luogo la riduzione dei canali comporta necessariamente una perdita di ridondanza del sistema e quindi non è ovunque possibile. Inoltre se da una parte si hanno forti vantaggi, sia economici che progettuali, nella riduzione dei canali, dall'altra l'elettronica sulla camera richiede l'uso di tecnologie resistenti alle radiazioni.
L'altro compito dell'elettronica del rivelatore, cioè l'assegnazione al segnale proveniente dal canale logico del rispettivo numero di bunch crossing, richiede un processo d'allineamento temporale: infatti, segnali provenienti da diversi canali logici, compiono percorsi diversi all'interno del rilevatore e vanno sincronizzati tramite dei ritardi programmabili prima di essere inviati in uscita.
La formazione dei canali logici, partendo dai 120000 fisici in uscita dagli ASD, è realizzata usando il chip DIAOLOG (DIagnostics time Alignment end LOGics), appositamente realizzati per l'esperimento, che ,tra le sue funzioni, ha quella di associare ad ogni canale logico un ritardo programmabile permettendone l'allineamento temporale. [Ref.19]

Anche se la struttura del rilevatore permette di disporre le schede di front-end (FEB) in gran parte del sistema, nelle regioni R3 e R4 delle stazioni M2-M5 e nella regione R2 di M4 e M5 si ha la necessità di unire canali fisici, provenienti da differenti FEB e da differenti camere, per formare i canali logici. Questo lavoro è svolto dalle schede IB (Intermediate-Board).
Una volta generati, i canali logici sono inviati alle schede ODE (Off-Detector Elettronics), dove gli viene assegnato il BX identifier (bunch crossing identifier) corrispondente per poi essere indirizzati al trigger di livello 0.
Un altro elemento importante dell'elettronica del rilevatore è il sistema di controllo ECS (Experiment Control System). L'ECS è basato su un sistema di comunicazione CAN bus ed ha lo scopo di comandare e monitorare le schede ODE e l'elettronica di front-end. Questo compito è svolto da schede appositamente progettate, dette Service Boards (SBs).
Parte del mio lavoro di tesi si è svolto nell'ambito della realizzazione del firmware delle SBs, nonché sul test delle stesse e delle loro funzionalità, in particolar modo nell'ambito del controllo delle schede DIALOG(Diagnostic, time Adjustment and LOGics).
Nel paragrafo successivo sarà data una breve descrizione delle singole strutture elettroniche menzionate, soffermandosi maggiormente sull'interfaccia ECS, sulle Service Boards e sulle loro funzionalità.

4.1 L'elettronica di Front-End

L'elettronica di front-end è situata all'interno del rivelatore stesso ed è costituita da due chip l'ASD e il DIALOG.
L'ASD (Aplifier-Shaper-Discriminator) riceve i segnali provenienti dalle camere e fornisce in uscita i canali fisici usando lo standard LVDS[1]. I segnali d'uscita

[1] Low Voltage Differential Signaling. E' un sistema di comunicazione differenziale basato cioè sulla differenza di potenziale tra due canali di trasmissione. E' un sistema studiato per dare un'alta velocità di trasmissione, dell'ordine dei Mbps, con una bassa dissipazione di potenza e poco rumore.

hanno una durata compresa tra i 50±10ns dipendente dalla forma del segnale in ingresso all'ADS. Ogni ASD chip presenta otto uscite verso le schede DIALOG.

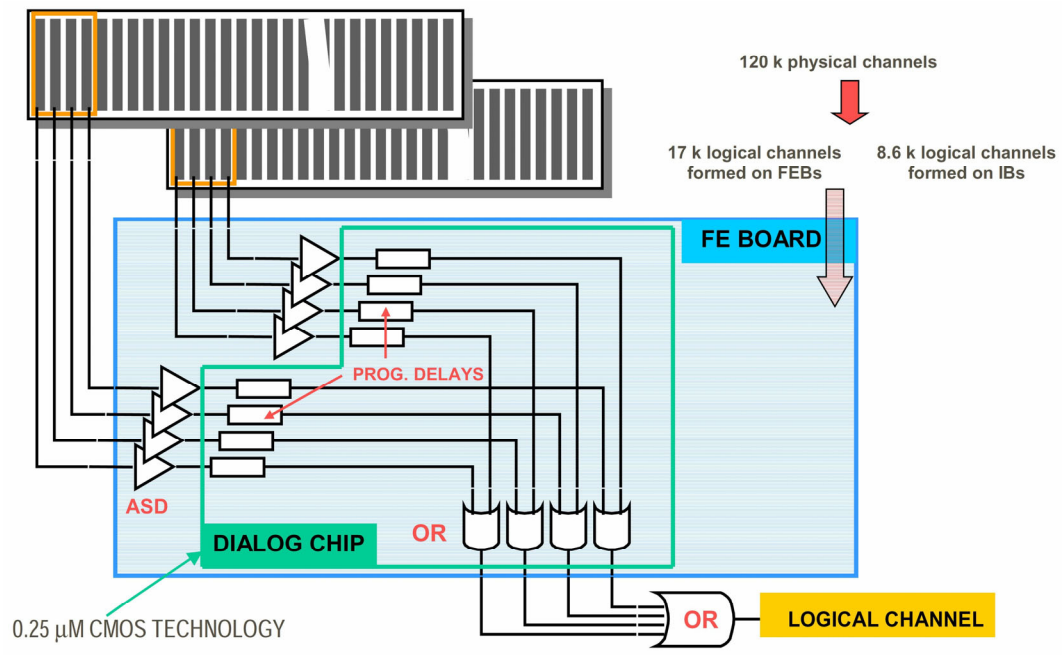


Figura 4.2: diagramma dell'elettronica di FE. Le schede ASD ricevono l'uscita delle camere formando i canali fisici (digitali) che vengono inviati al DIALOG chip (fino a due ASD per ogni DIALOG). Qui i canali fisici sono allineati temporalmente e messi in or per andare a formare i canali logici.

Il DIALOG (Diagnostic, time Adjustment and LOGics) è un chip custom sviluppato secondo la tecnologia "IBM 0.25 µm rad-hard" che fornisce una gran tolleranza alle radiazioni.

Per conservare la modularità ad otto, ogni chip è dotato di 16 ingressi LVDS, così da poter servire fino a due ASD. Il suo compito principale è quello di allineare e mettere in OR logico i canali fisici per formare, in uscita, i canali logici.

Ogni segnale è inviato ad un ritardo programmabile dove può essere ritardato fino a 25ns in passi da 1,5ns. Ognuno dei 16 ingessi LVDS può essere mascherato singolarmente tramite un registro a 16bit, in modo da permettere di individuare un

singolo canale fisico per individuare eventuali malfunzionamenti. Il segnale viene poi modellato dal "digital shaper" ad una lunghezza inferiore a 25ns in modo da non sovrapporsi con quelli provenienti dal successivo bunch crossing. Infine più segnali sono messi in OR ottenendo fino ad otto Canali logici che vengono inviati alle schede ODE o alle IB. In particolare è possibile avere otto, quattro o due canali logici in uscita, a seconda della zona del rilevatore in cui ci si trova.

Ogni singolo passo dei ritardi e dei digital shapers è regolato da un DLL[2] che viene calibrato all'accensione tramite un clock proveniente dall'ECS.

Nella figura 4.3 è riportato il diagramma di flusso del DIALOG.[Ref 20].

Come mostrato è anche possibile esaminare un canale fisico singolarmente, selezionandolo tramite un Multiplexer, inviandolo ad uno scaler che ne può contare gli hits per una frazione di tempo regolabile accumulandoli in un contatore leggibile dall'esterno tramite ECS.

Nel DIALOG sono implementati anche due DAC che servono per generare e controllare indipendentemente le soglie dei due chip ASD. Queste soglie possono essere regolate, tramite un registro ad otto bit, in un range tra 0 e 2,5 Volt.

Il controllo e il monitor del chip sono eseguiti dall'ECS tramite un'interfaccia i2c che permette la scrittura e la lettura di tutti i registri.

Ogni singola FEB serve 16 canali fisici e consiste in un DIALOG e due ASD. È previsto l'uso di 7536 FEB.

[2] Delay Locked Loop. Circuito integrato a reazione negativa che mantiene costante il ritardo tra la frequenza del segnale di ingresso e quella generata dal clock interno

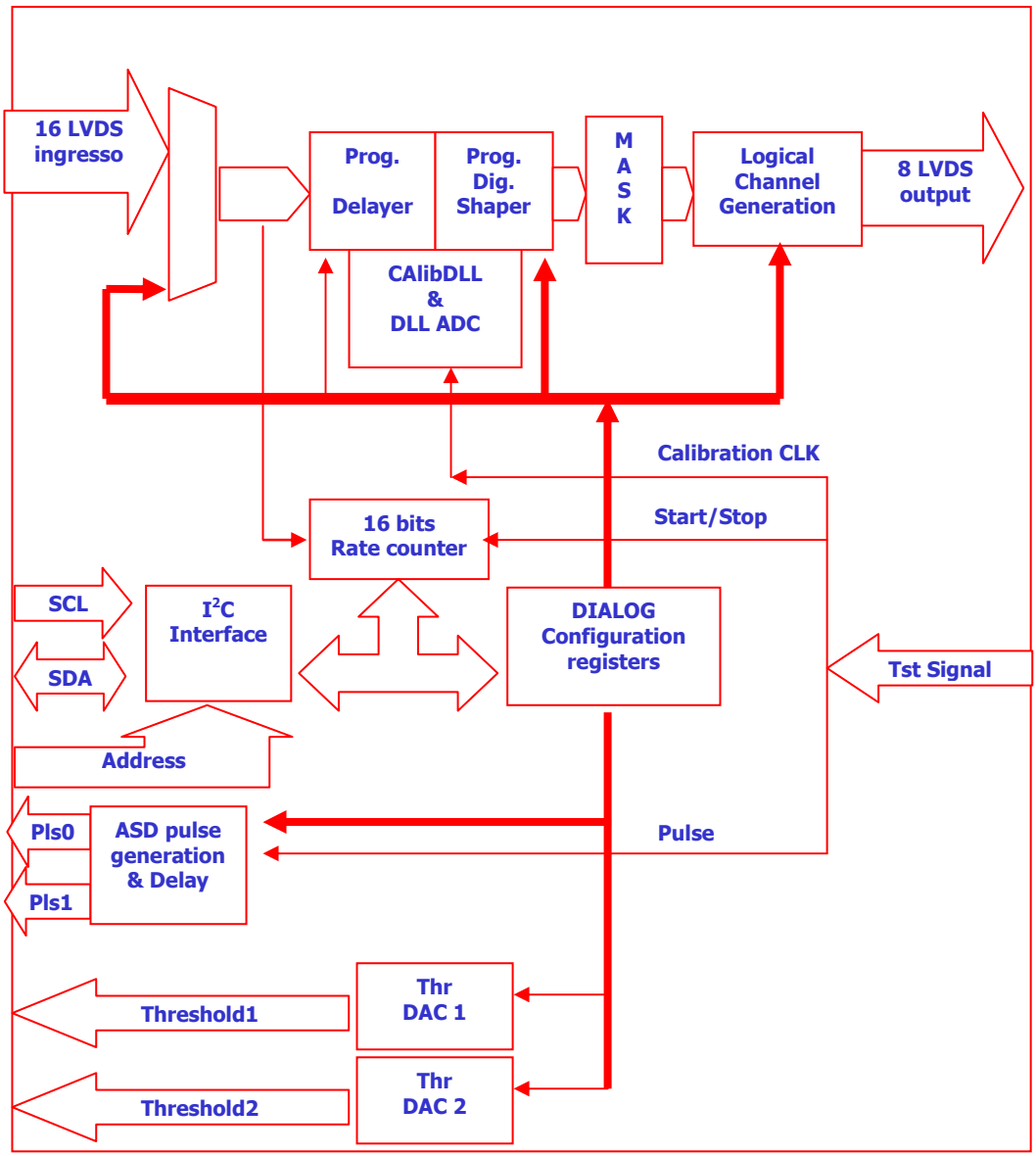


Figura 4.3: Diagramma del chip Dialog.

4.2 Le Intermediate-Board (IB).

Le IB, situate sul lato del rivelatore, completano la formazione dei canali logici fornendo un ulteriore livello di or logici. Questo è necessario perché alcuni canali

logici sono formati con canali fisici provenienti da camere diverse, in particolare dalla regione R2 di M4 e M5 e dalle regioni R3 e R4 di M2-M5.
Le IB sono schede molto semplici, composte di soli OR logici, e supportano fino ad un massimo di 192 ingressi e 64 uscite in standard LVDS.

Nr logical/physical	M1	M2	M3	M4	M5
R1	576/576	336/336	336/336	288/288	288/288
R2	576/1152	384/672	384/480	168/288	168/288
R3	576/1152	336/1152	336/1152	120/576	120/576
R4	576/1152	336/1152	336/1152	120/1152	120/1152

Tabella 4.1: rapporto tra canali logici e canali fisici nei quadranti di ogni camera. Si noti che la riduzione maggiore dei canali avviene nelle zone interessate dalle IB.

L'uscita delle IB, insieme ai canali logici già formati nei DIALOG, costituiscono l'ingresso per le schede ODE.

4.3 Le schede ODE(Off Detector Electronics).

Il compito delle schede ODE è quello di sincronizzare il segnale e renderlo disponibile per il trigger di livello 0. [Ref.21]
Anche le pipeline di L0 sono inclusi in questa parte dell'elettronica.
Le ODE sono dotate di 192 ingressi LVDS e dispongono di 13 link ottici a 1,6 Gbit/s: 12 per comunicare con il trigger di livello 0 e uno per trasmettere i dati opportunamente impacchettati alle schede TELL1 (Trigger ELectronics and L1

board) [Ref.20] esterne che contengono il buffer per il trigger di livello 1 e il sistema di acquisizione dati DAQ. La comunicazione con il sistema di controllo ECS è gestita da un modulo ELMB interno su interfaccia CAN.

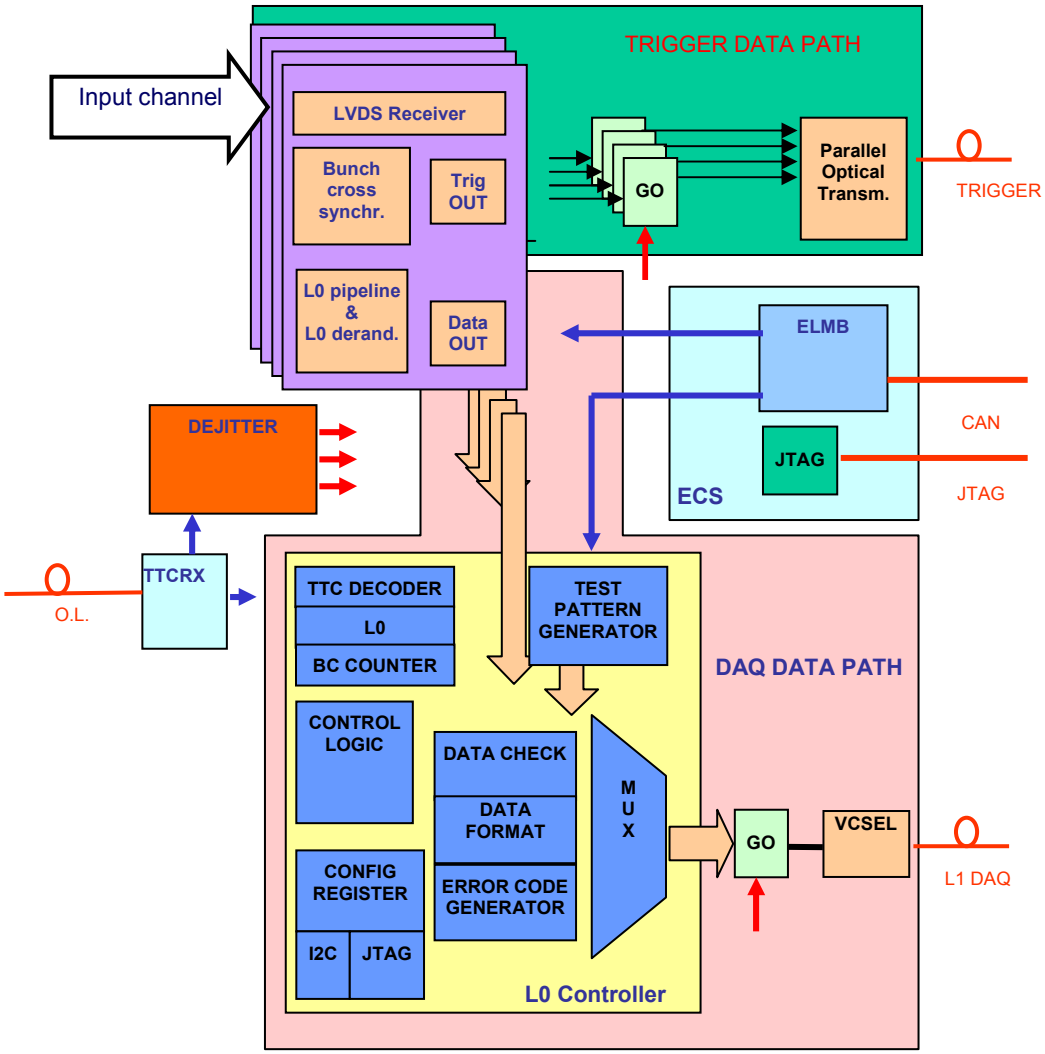


Figura 4.4: Diagramma di funzionamento delle schede ODE.

La sincronizzazione del segnale avviene tramite 24 chip custom ad otto bit, detti "Sync chip", che contengono dei receivers LVDS, le pipeline di L0 e il "derandomizer". Tredici GOL (Giga bit Optical Transmitter) , chip custom progettati con tecnologia resistente alle radiazioni, gestiscono invece le comunicazioni su link ottico.

Le schede ODE hanno le dimensioni di un modulo 6U-VME e sono contenute in dei crate situati nel tunnel del rivelatore, a qualche metro da esso: ogni crate contiene 18 schede.

In figura 4.4 mostra lo schema di funzionamento delle schede ODE: al segnale in ingresso viene assegnato il giusto BX identifier e viene inviato alle pipeline per L0. Parallelamente il dato e i quattro bit meno significativi del BX identifier vengono inviati al trigger di livello 0. I dati nelle pipeline attendono la decisione di quest'ultimo e, in caso di risposta affermativa, vengono trasmessi attraverso link ottico alle schede TELL1. Qui i dati rimangono nel buffer di L1 in attesa della decisione del trigger di livello 1 e, se anche questa è affermativa, il codice viene messo a disposizione per il DAQ.

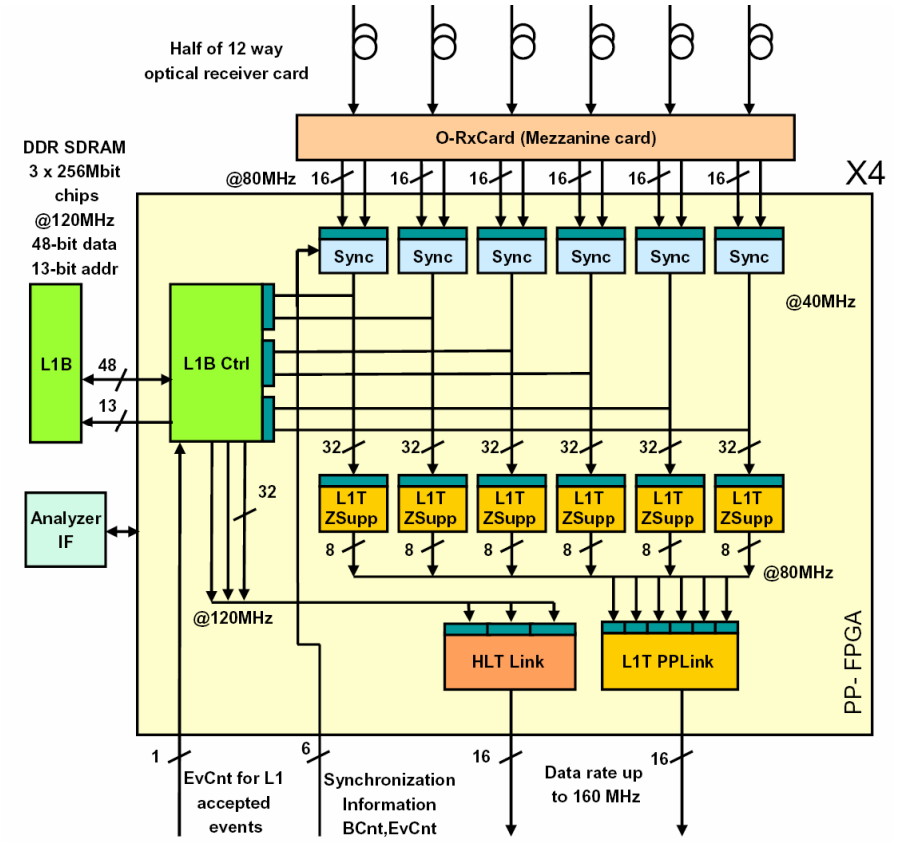


Figura 4.5: Schema a blocchi di una scheda TELL1.I dati provenienti dalle schede ODE vengono messi nel buffer di L1 su DDR SDRAM. Parallelamente vengono zero-suppressed e spediti al trigger di livello 1 (L1). Quelli che ricevono risposta affermativa da quest'ultimo vengono inviati al DAQ.

4.4 L'ECS

L'ECS, Experiment Control System, è un sistema di controllo che ha lo scopo di monitorare e controllare le funzioni del rivelatore di muoni.
L'architettura di questo sistema è basata sull'ELMB (Embedded Local Monitor Box), una scheda con nodo CAN progettata in collaborazione con ATLAS per lavorare in ambienti a moderato livello di radiazioni.[Ref 22].
L'ELMB (Embedded Local Monitor Box) è una piccola scheda plug-on basata su un microcontrollore commerciale a 8 bit ATmega128 e un CAN-Controller.

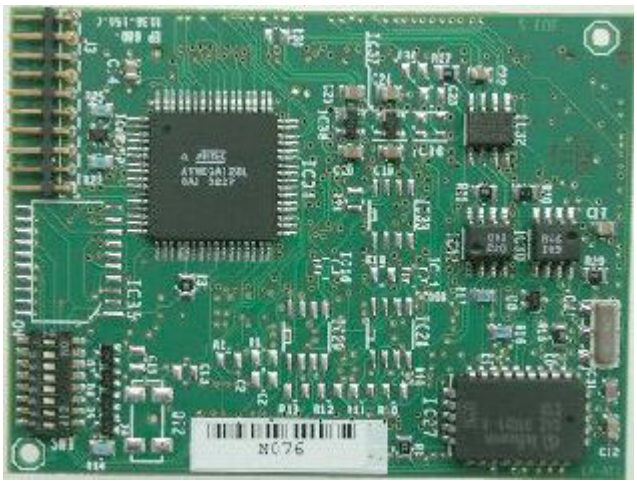


Figura 4.6: la scheda plug-on ELMB con microcontrollore ATmega128.

Il microcontrollore dell'ELMB è un processore a 8 MHz che dispone di 128 kbytes di memoria flash per la scrittura del firmware contenente il bootloader con funzioni di "watch-dog" che riavvia l'ELMB in caso di SEU[3] (Single Event Upset).(Cap. 5.1.1)

[3] Si parla di errore SEU quando il passaggio di una particella carica all'interno del silicio provoca la commutazione di uno o più transistor cambiando lo stato dei flip-flop di una SRAM, provocando la perdita di dati e, il più delle volte, un'interruzione del funzionamento dell'applicazione in uso.

Gli ELMB sono raggruppati in differenti rami CAN bus ognuno contenente 24 nodi CAN .

La struttura dell'ELMB sarà approfondita nei paragrafi successivi.

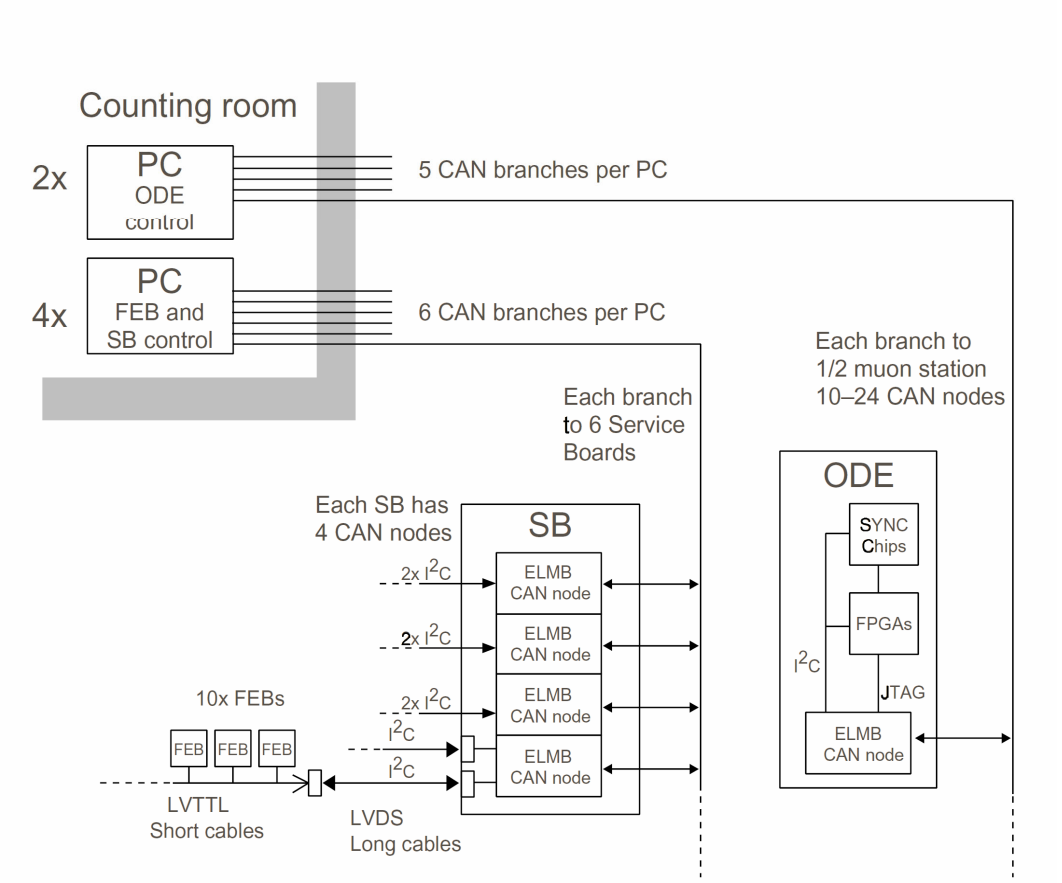


Figura 4.7: Il sistema ECS (Experimet Control System) per le camere per i muoni. Si possono vedere l'interfaccia CAN ECS per le Service Boards (SB) con le numerose uscite i2c (12 per ogni SB) verso le schede di Front-End (FEBs) e i rami CAN per le schede ODE.

L'ECS è diviso in due sottosistemi: uno, con 24 rami CAN bus, che controlla l'elettronica di Front-End attraverso le SB, l'altro, con 10 rami CAN bus, controlla le schede ODE.

I rami CAN sono utilizzati tramite 6 pc su cui sono montate delle schede "PCI-CAN interface" commerciali con 2 nodi CAN ognuna, così suddivisi: 4 pc con 3

schede PCI-CAN, per un totale di 24 nodi CAN, per i 24 rami delle SB; 2 PC con 2 schede PCI-CAN che si occupano del sottosistema ECS-ODE. [Ref.18]

4.4.1 LA SERVICE BOARD

La Service Board (SB) [Ref. 23] è una scheda nelle dimensioni standard 6U VME, che alloggia 4 schede ELMB. Ogni ELMB può servire tre i2c bus per lunga distanza e uno locale.

Tali bus possono essere utilizzati fino ad una distanza di 10 metri usando dei drivers LVDS (Low Voltage Differential Signalling) e vengono chiamati "long i2c branches". I rami i2c si estendono fino alla camera per i muoni, dove, tramite un receiver LVDS, possono servire fino a 10 DIALOG [Ref. 20].

I compiti della Service Board sono principalmente il controllo dell'elettronica di Front End, in particolar modo del DIALOG, e l'individuazione e la correzione d'eventuali errori nei suoi registri. Tramite i bus i2c, può accedere a tutti i registri interni del chip DIALOG, permettendone la scrittura e la lettura. Inoltre periodicamente l'ELMB ne testa la consistenza e, nel caso in cui sia rilevato un errore, lo segnala all'interfaccia CAN e provvede al refresh del codice ivi contenuto. Un'altra funzione della Service Board è quella di gestire la comunicazione tra l'interfaccia CAN-bus dei pc e il DIALOG, rendendo possibile, ad esempio, la lettura dei contatori o l'immissione dei valori di soglia, nonché la calibrazione della DLL. Dispone, inoltre, di un chip FPGA della Actel dove sono implementate funzioni di fondamentale importanza per il controllo e il test dell'apparato. Infine manda e ricevere segnali utili per la calibrazione di tutta l'elettronica di front-end e per monitorare che questa e le camere stanno operando nelle giuste condizioni.

Le Service Board sono disposte sul lato del rivelatore, tra le IB e le schede ODE. Sono necessarie un totale di 144 Service Board.

Ci sono due modi per implementare le funzionalità delle SB: è possibile creare dei task nell'ELMB stesso che compiano le operazioni volute, quando richiamate tramite comando proveniente dal CAN-bus, oppure si possono utilizzare direttamente dal PC-CAN le istruzioni i2c. Il primo di questi metodi è da preferirsi perché semplifica notevolmente le comunicazioni tra apparato e PC-CAN e permette di ottenere prestazioni performanti. Esisteranno inoltre altri task che, indipendentemente dall'esterno, monitoreranno il corretto funzionamento del sistema fornendo eventualmente segnali d'errore sul nodo CAN.

Parte del lavoro di tesi si è svolto nell'ambito dell'implementazione e del test, di alcuni di tali task, nel firmware dell'ELMB in modo da permettere un'automatizzazione delle funzioni di monitoraggio e controllo del chip DIALOG e della Service Board stessa.

Nei paragrafi successivi verrà descritto più in dettaglio l'hardware e il firmware della Service Board e il lavoro da me svolto.

5. La struttura della Service Board (SB)

5.1 L'hardware

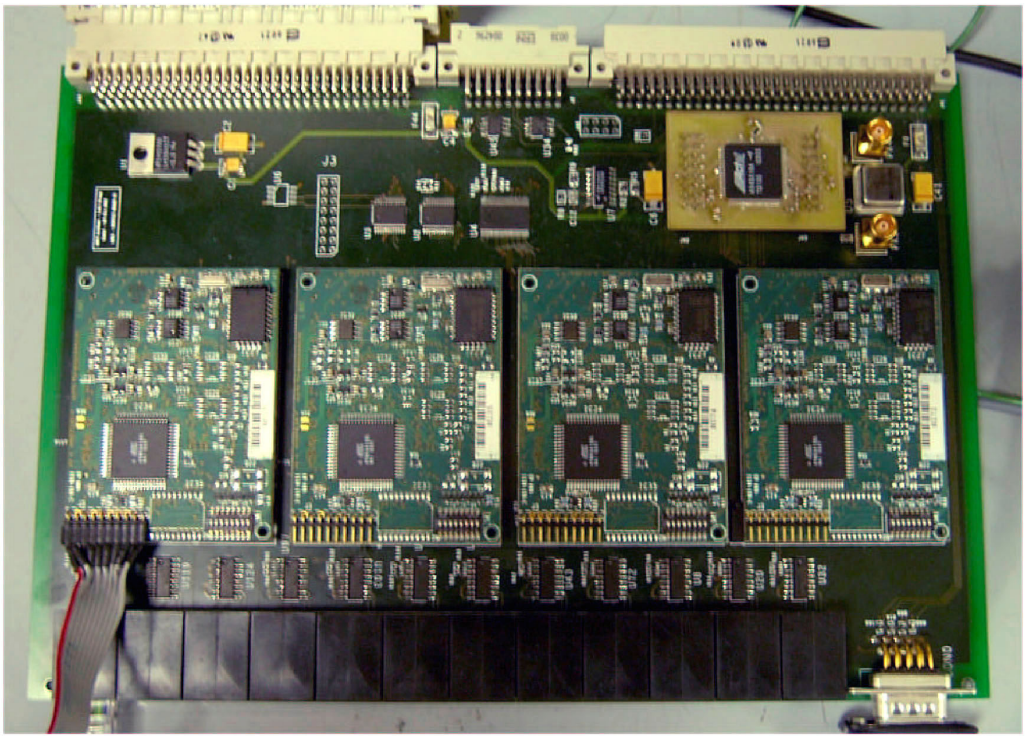


Figura 5.1: La Service Board

La Service Board consiste in 4 schede ELMB e nell'insieme dei componenti che le consentono la comunicazione con l'elettronica di front-end e lo svolgimento dei suoi compiti. Può comunicare con i "PC-CAN interface" tramite un CAN-bus pilotato dall'ELMB stesso, mentre dispone di un bus interno i2c, nello standard TTL, per la comunicazione interna.

Sempre tramite il bus i2c pilota 12 "link LVDS" (tre per ogni ELMB) che permettono la comunicazione con l'elettronica di front-end. [Ref.24]

Le Service Board sono inserite all'interno di un "Service ECS crate", in blocchi da 16, insieme ad un modulo PDM (Pulse Distribution Module) che distribuisce clock a 40MHz di LHC, ricevuto dai moduli TTCrx, e il segnale impulsivo periodico "BC_pulse", ottenuto ritardando e dividendo tale clock.
Le Service Board d'ogni crate sono divise in 4 canali CAN-bus, ognuno composto di 4 schede, per un totale di 16nodi CAN. Il controllo dei canali è effettuato tramite una scheda "Kvaser PCI-CAN bus" montata su un calcolatore esterno al rivelatore.

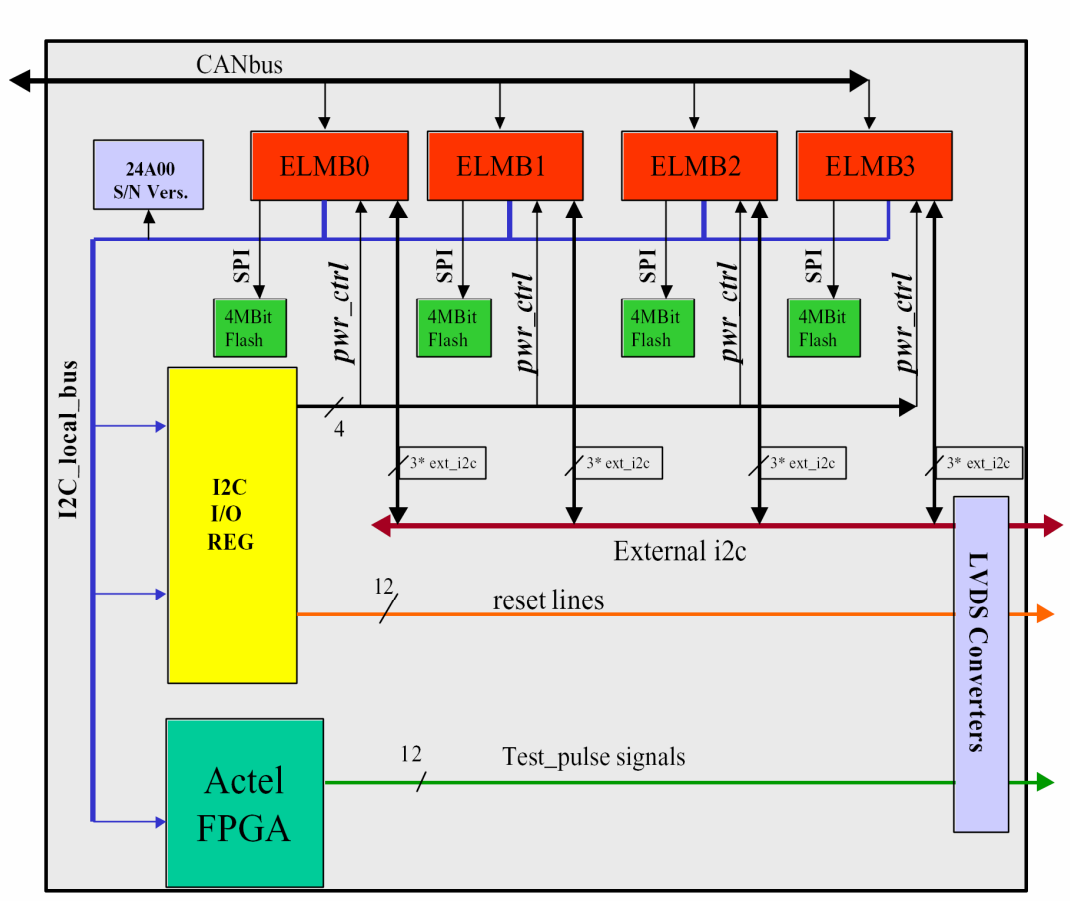


Figura 5.2: Diagramma a blocchi della Service Board.

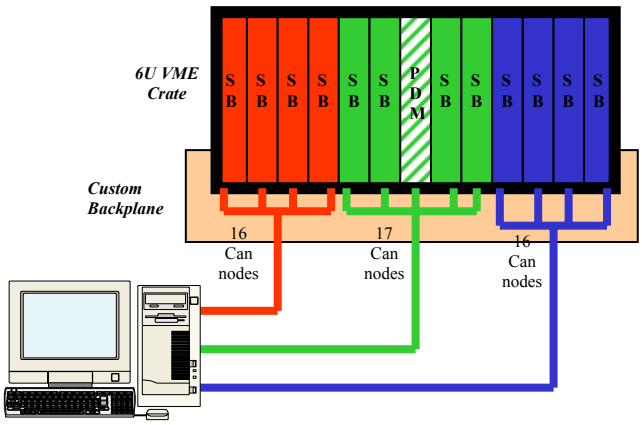


Figura 5.3: Crate 6U-VME con le SB divise in gruppi da 4 e la scheda PDM contenente il chip TTCrx.

La figura 5.2 rappresenta il diagramma a blocchi della ServiceBoard, evidenziandone i componenti principali. Come si può osservare su un unico canale CAN sono posti quattro moduli ELMB, di cui parleremo in un paragrafo dedicato, ognuno dotato di 4 Mbit di memoria Flash con la quale comunica tramite interfaccia SPI. La memoria flash è particolarmente resistente ad errori dovuti a radiazioni e questo le rende immuni a "Single Event Upset" (SEU).

Lungo il bus i2c interno troviamo un EEPROM, che contiene il numero di serie della Service Board, due registi remoti di I/O a 16bit "philips PCF8575" e una FPGA (Field Programmable Gate Array) della ACTEL.

Infine sia le ELMB, sia l'ACTEL, sia i registri di I/O, comunicano con l'elettronica di front end tramite 12 "i2c external links", tre per ogni ELMB, che necessitano della conversione del segnale i2c dallo standard TTL a LVDS (per le uscite) e viceversa (per gli ingressi).Questo è reso possibile da 12 lvds drivers DS90LV047ATM e di 4 lvds receivers DS90LV048ATM, entrambi i componenti tolleranti alle radiazioni.

5.1.1 L'ELMB

La scheda ELMB (Embedded Local Monitor Board) (fig.4.5) è il cuore della Service Board. Essa si occupa della gestione di tutte le comunicazioni CAN e delle comunicazioni i2c fra le varie parti della Service Board e dell'elettronica di front-end fungendo da i2c master.

In figura 5.4 è riportato il diagramma a blocchi dell'ELMB.

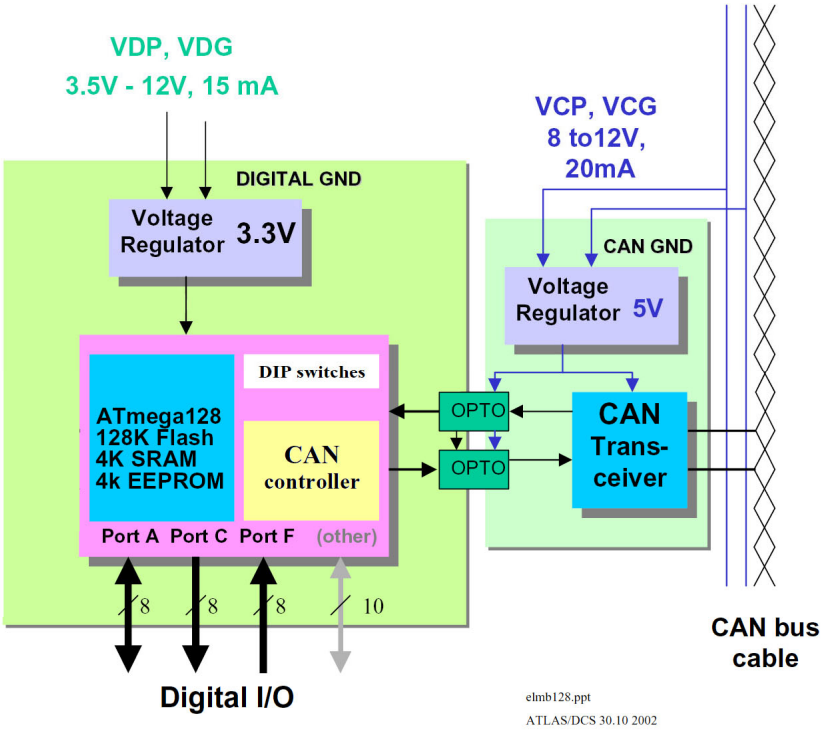


Figura 5.4: Diagramma a blocchi dell'ELMB.

L'ELMB è basato sul microcontrollore "ATmega128 AVR ATMEL", ad architettura RISC, con 121 istruzioni a clock singolo e 8 MHz di frequenza di clock [Ref.24].

Il microcontrollore ha a disposizione 128 kbytes di memoria flash per il firmware, 4 kbytes di EEPROM e 4 kbytes di SRAM.

Uno "SPI CAN controller SAE81C91" permette di comunicare dall'esterno con il microcontrollore accedendo al CAN-bus attraverso il "CAN-trasmitter/receivers" dell'ELMB, con cui è connesso da un link ottico che ne garantisce l'isolamento galvanico.

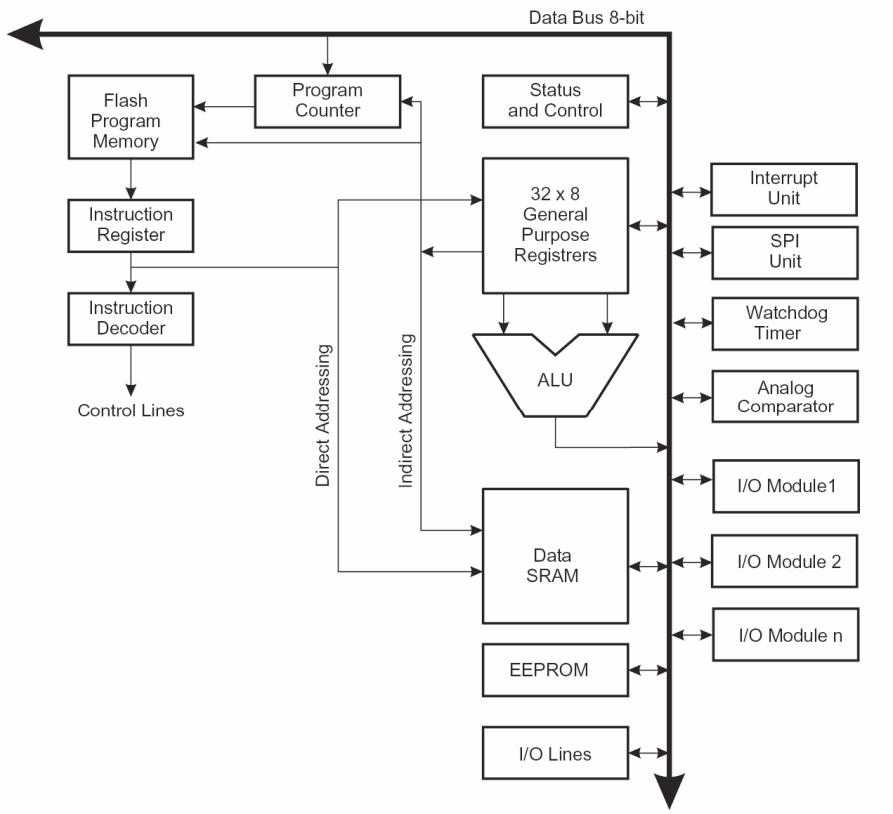


Figura 5.5: Architettura AVR dell'ATmega128

L'ATmega128 è collegato al "DIP-switches" dell'ELMB che contiene l'indirizzo interno della scheda, detto NodeID, a cui bisogna riferire i comandi perché siano univocamente indirizzati ad uno degli ELMB. È possibile configurare l'indirizzo interno agendo direttamente sul DIP-switch attraverso l'interruttore evidenziato in figura 5.7 . L'ELMB DIP-switch contiene anche le impostazioni sulla velocità di trasmissione e abilita, o disabilita, la programmazione via CAN. È possibile, infatti, programmare il microcontrollore sia tramite un apposito connettore, utilizzando il software PonyProg2000 e un apposito programmatore, sia impartendo istruzioni direttamente attraverso il nodo CAN.

La memoria flash dell'ATmega128 è divisa in due parti: il "BootProgram" e l' "Application program". Nella prima parte viene scritto il "bootloader" che contiene le operazioni di inizializzazione del chip e il programma di avvio. I registri di questa parte di memoria dispongono di bit di protezione per la scrittura e la lettura che ne evitano la sovrascrittura accidentale. La seconda contiene

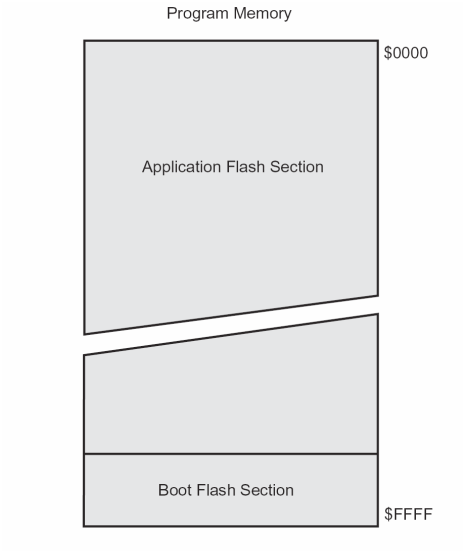


Figura 5.6: Mappa della memoria flash dell'ATmega128

invece il programma principale. Le dimensioni delle due sezioni di memoria sono variabili una a discapito dell'altra.

Il firmware dell'ELMB è composto di due elementi: il "bootloader" e il programma principale [Ref.25]. All'accensione dell'hardware il microcontrollore legge il bootloader che, da solo, permette di scrivere o modificare il firmware attraverso CAN. Dopo 4 secondi controlla se, nella parte di memoria flash "Application program", risiede il programma principale e, se è presente, l'esegue. Se non è presente, invece, il microcontrollore resta in attesa finché non è attivato da un messaggio CAN o da una scrittura del programma principale, la cui presenza viene rilevata ogni 4 secondi.

In caso di chiamata da parte di una subroutine o di un interupt i dati necessari al ripristino del programma principale vengono salvati nello stack. Per permettere una rapida esecuzione di questi salti, lo stack viene allocato nella memoria SRAM a disposizione del microcontrollore che non è tollerante alle radiazioni e soggetta a errori di tipo SEU. Il deterioramento dei dati contenuti nello stack rende impossibile il ripristino del programma principale e provocando un arresto del sistema. Per ovviare a questo inconveniente il bootloader fornisce anche un sistema "watch-dog"[4] per la rivelazione d'eventuali errori di tipo SEU: in caso d'errore provvede a riavviare l'hardware ripristinando nuovamente il codice dalla memoria flash.

[4] Si parla di sistema "watch-dog" (letteralmente "cane-arrabbiato") per la rivelazione degli errori quando il mancato invio di un messaggio di corretto funzionamento, da parte del software monitorato, entro un tempo stabilito provoca il riavvio del sistema.

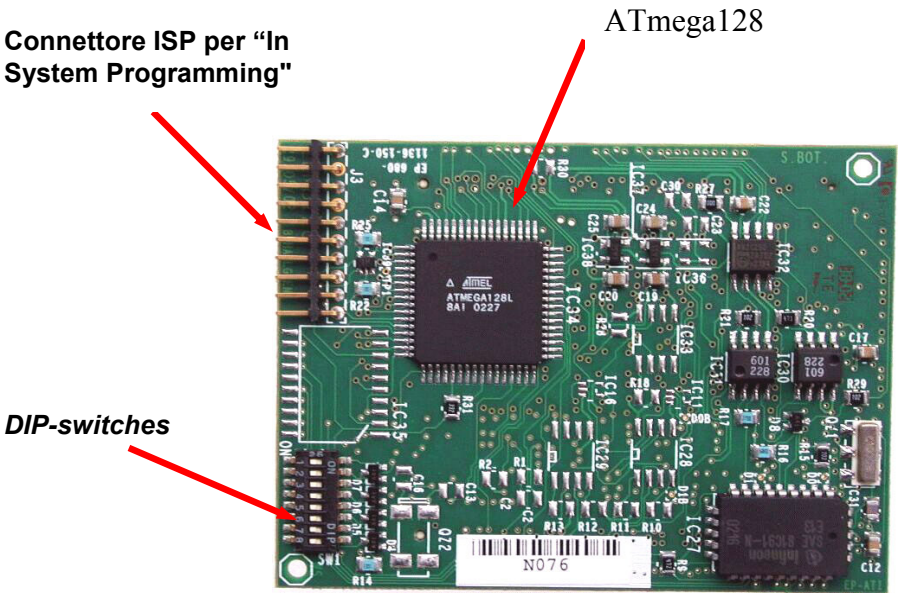


Figura 5.7: Facciata superiore dell’ELMB.

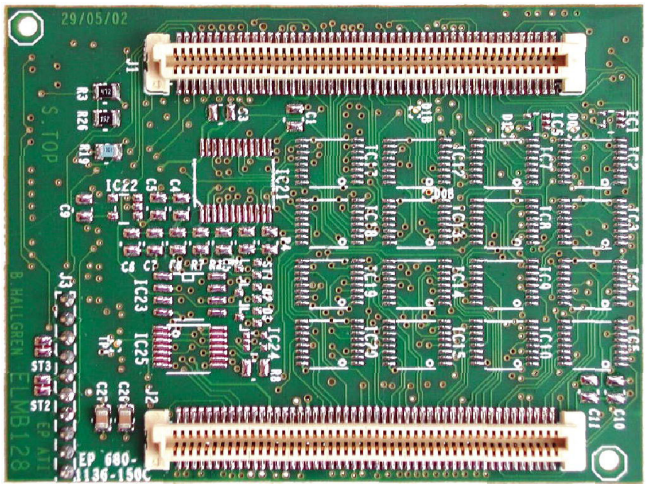


Figura 5.8: Facciata inferiore dell’EMB.

5.1.2 *SPI FLASH ROM*

Il microcontrollore dell'ELMB dispone di una Serial Peripheral Interface (SPI). Tale interfaccia permette un rapido trasferimento di dati tra microcontrollore e periferiche. Inoltre la Service Board dispone di 4 Mbit di memoria Flash con interfaccia SPI per ogni ELMB. Questo tipo di memoria è pressoché immune al SEU e, quindi, si presta a contenere dati importanti come le soglie e i parametri di riferimento dell'elettronica di front end.

5.1.3 EEPROM

Un modulo d'identificazione su EEPROM (Microchip 24AA00), anch'esso tollerante alle radiazioni, contiene il numero di serie dell'Service Board che ne permette una rapida identificazione necessaria per il corretto funzionamento dell'ECS.
Contiene, inoltre, dati relativi alla versione del firmware e dell'hardware.

5.1.4 I/O REGISTERS

I registri di I/O sono forniti da due chip "Philips PCF8575" [Ref 26].Questo chip è provvisto di un'interfaccia i2c bus, tramite la quale l'ELMB può accedere a 16-bit di I/O per ogni chip, che controllano le 12 linee di reset dell'elettronica di front end, le 4 linee per lo spegnimento dei quattro moduli ELMB stessi e la linea di reset dell'ACTEL.
Nel firmware dell'ELMB sono state implementate scorciatoie d'accesso diretto alle linee di reset e alla funzione di spegnimento, protette da password, in modo da evitare che operazioni errate possano provocare reset involontario o spegnimento accidentale di una o più ELMB.

5.1.5 Actel FPGA

La Service Board è dotata di un chip FPGA "Actel AT54SXA16", dove è implementata la logica necessaria per il controllo della Service Board stessa e per la generazione di segnali impulsivi utili per il test e il funzionamento dell'elettronica di front-end.

Di seguito sono riportati lo schema dei registri interni e dei blocchi di logica dell'Actel e i segnali impulsivi che può generare.

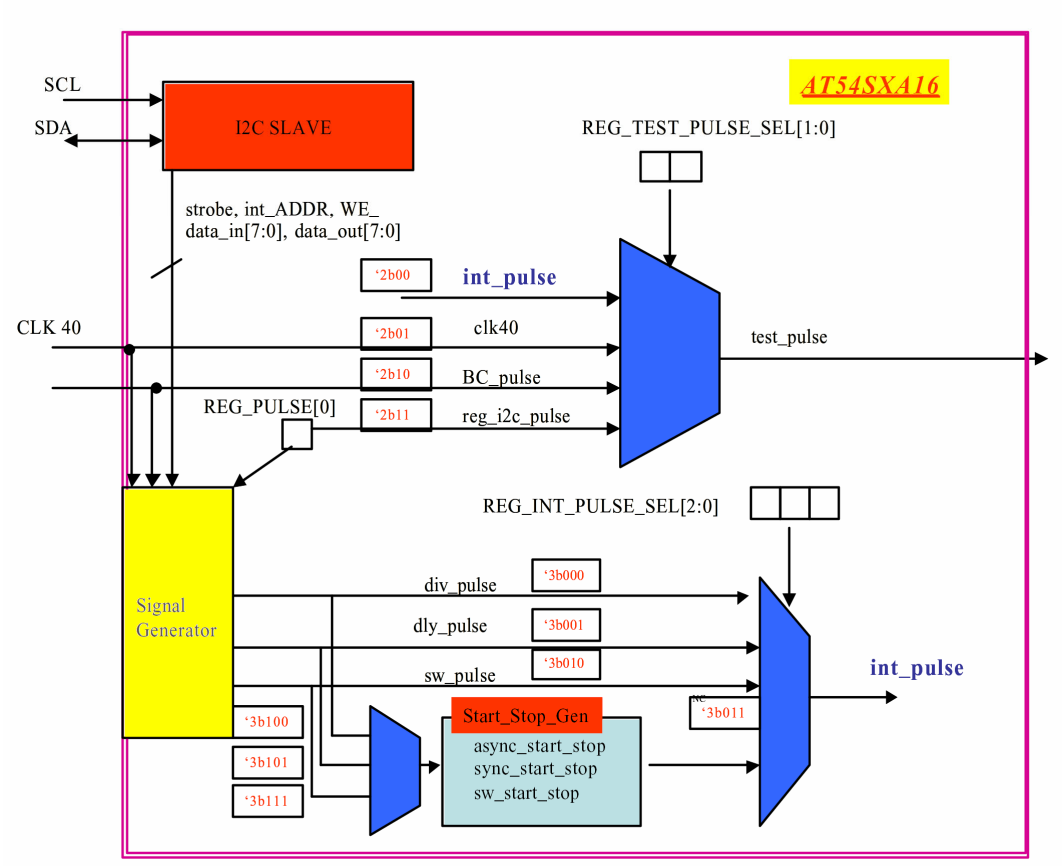


Figura 5.9: Diagramma a blocchi delle funzionalità implementate nell'FPGA Actel AT54SXA16

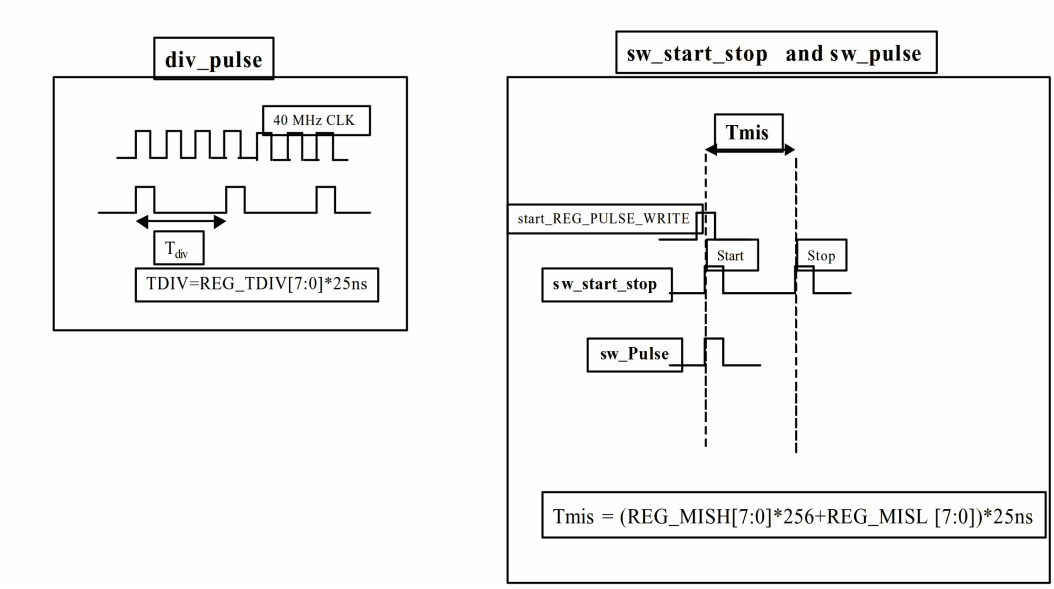


Figura 5.10a/b: I segnali div_pulse (a), sw_start_stop (b) e sw_pulse (b) prodotti dall'FPGA Actel.

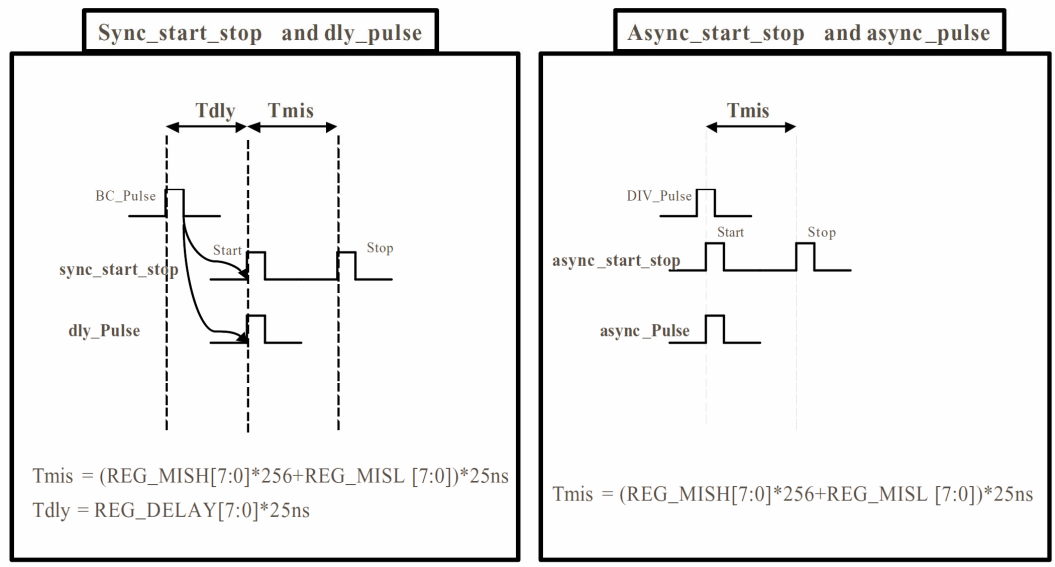


Figura 5.10c/d: I segnali Sync_start_stop (c), dly_pulse (c), Async_start_stop (d) e async_pulse (d) prodotti dall'FPGA Actel.

Come mostrato, all'interno dell'FPGA è implementata un'interfaccia i2c slave con cui accedere ai registri interni che determinano lo stato dei multiplexer che selezionano i segnali impulsivi.

È anche possibile selezionare il clock a 40 MHz di LHC, proveniente dalla Service Board, tra l'altro necessario per la generazione di tutti i segnali periodici. La progettazione dell'Actel AT54SXA16 è stata effettuata in verilog.
Durante il lavoro di tesi è stato implementato, nel firmware dell'ELMB, l'accesso a questi registri ed è stato creato, all'interno del programma di gestione del nodo CAN su PC remoto (programma client), un menu d'accesso alle funzioni dell'Actel. Inoltre ho partecipato alla fase di test e debugging della logica, che ha portato alla realizzazione di una seconda versione. In particolare è stato sostituita l'interfaccia "i2c slave" asincrona con una sincrona poiché la prima era soggetta al rumore prodotto dal clock interno. Sono stati anche aggiunti 2 nuovi registri, che permettono il mascheramento dei 12 canali d'uscita, e ulteriori 4 registri, accessibili in sola lettura, contenenti la versione dell'FPGA e la sua data di scrittura.
Nel paragrafo dedicato al firmware sarà descritto in maniera dettagliata il lavoro svolto e il risultato dei test mettendo in luce le funzionalità dell'actel.

5.1.6 Protocollo i2c-like LVDS

La comunicazione tra la Service Board e l'elettronica di front-end (DIALOG) è resa possibile tramite cinque segnali: quattro coppie di segnali LVDS ed una line di reset.
I dati sono trasmessi grazie all'interfaccia i2c, su una versione leggermente modificata dell'i2c-bus. Le FE board funzionano come periferiche i2c slave e hanno a disposizione una linea unidirezionale SCL e due linee, una d'ingresso e una d'uscita, per SDA.

5.1.7 Il modulo PDM.

Il modulo PDM (Pulse Distributor Module) è realizzato su una scheda nello standard 6U-VME ed ha il compito fondamentale di rendere disponibile al sistema di controllo dell'esperimento (ECS) il clock di LHC. Questa funzione è svolta tramite un chip TTCrx [Ref. 27] e un ELMB che si occupa della sua gestione. Il clock di LHC è distribuito in tutto il rivelatore LHCb tramite il sistema TCF (Timing and Fast Control system) [Ref.28] che costituisce la soluzione alla necessità di avere, per tutta l'elettronica, un clock in fase. Inoltre fornisce un'interfaccia di comunicazione in broadcast ed una indirizzabile per tutto l'apparato. Un'altra funzione del modulo PDM è quella di ricevere e gestire i messaggi trasmessi lungo queste linee.

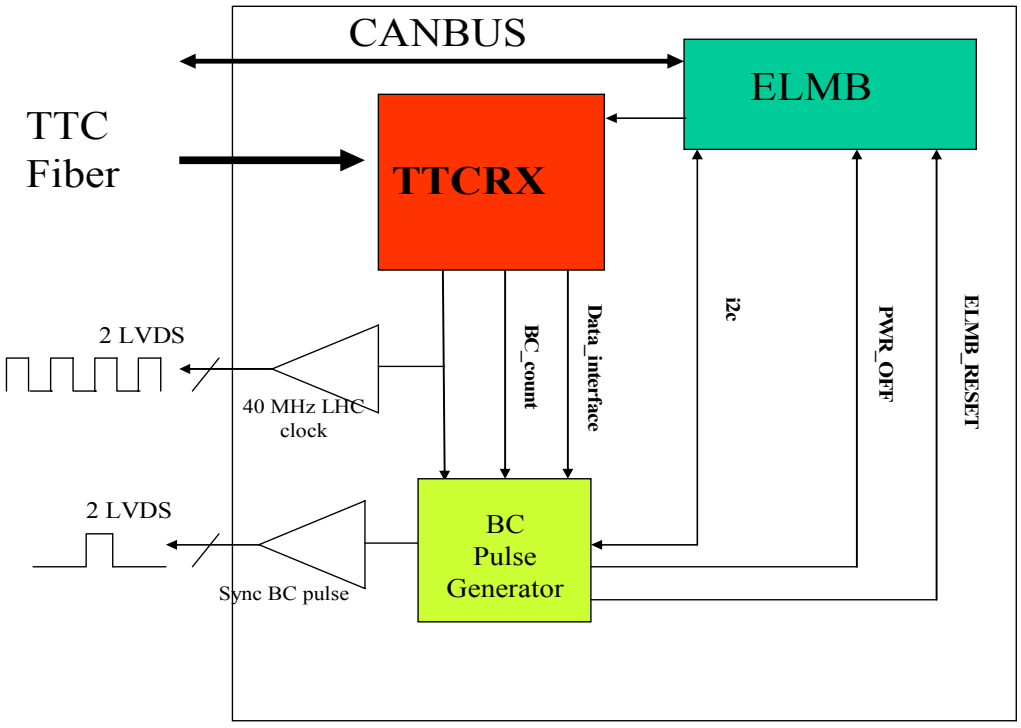


Figura 5.11: diagramma a blocchi del modulo PDM.

Il TTCrx è anche utilizzato per generare il segnale di test BC_pulse, fondamentale per il debugging di tutto l'apparato, che viene prodotto dividendo il clock ricevuto e risulta in fase con quest'ultimo. Si noti che la coerenza di fase sarà di fondamentale importanza ogni volta che varrà usato il segnale BC_pulse come riferimento, come in molti dei test.
Nella scheda PDM è stata introdotta una linea di comunicazione, fra il TTCrx è il modulo FPGA Actel, che genera il BC_pulse dal BC_count, in modo da permettere il reset e lo spegnimento, dell'unica ELMB presente, comandati dall'interfaccia di comunicazione indirizzabile del sistema di distribuzione del clock dell'apparato.

5.2 Il firmware

5.2.1 CANOpen

Il CAN-bus è completamente gestito, tramite il controllore CAN, dal bootloader originale dell'ELMB [Ref.25], semplificando notevolmente il lavoro necessario nella scrittura del firmware.
Ad un livello più alto, però, è necessario stabilire secondo quale protocollo utilizzare i nodi CAN.
Si è optato per l'utilizzo del protocollo CANOpen [Ref 29].
Il CanOpen assume che le periferiche dispongano di un transmitter-receivers e un controller CAN nelle specifiche ISO 11898.
La parte più importante del protocollo CanOpen è la definizione degli oggetti nell' "object dictionary". Praticamente è necessario definire degli oggetti accessibili via network in modo tale da poterli richiamare con un determinato indirizzo. Ogni oggetto può rappresentare o una funzionalità implementata nel firmware stesso, o un protocollo di trasmissione dati.
Possiamo classificare gli oggetti nel modo seguente:

- Process Data Objects (PDOs), tramite i quali si eseguono i trasferimenti di dati in real-time

- Service Data Objects (SDOs), che gestisce gli accessi in scrittura e lettura dei registri hardware della Service Board, come i registri di I/O o l'Actel.

- Special Function Object, che gestisce la sincronizzazione del network, alla trasmissione d'eventuali messaggi d'errore e alla caratterizzazione temporale.

- Network Management (NMT), che fornisce i servizi per l'inizializzazione del network, il controllo degli errori e dello stato dell'hardware.

5.2.2 SB Service Data Object (SDO)

L'SDO merita un'attenzione particolare perché in essa sono definite le funzioni di accesso all'hardware e gli oggetti propri della Service Board, oltre a quelli standard dell'ELMB CAN-bus.
Parte del lavoro di tesi si è svolto nell'ambito del perfezionamento di tale dizionario, introducendo gli oggetti 4500 e 4501 per il controllo dell'accensione dell'ELMB, 4600 e 4601 per il controllo delle linee di reset dell'elettronica di front-end, 4700 per l'accesso ai registri dell'Actel e 5000-5299 per l'accesso hai registri del DIALOG chip.
Nella tabella 5.1 sono riportati gli indirizzi di tutti oggetti dell'SDO.
Gli indirizzi 4000,4100 e 4200 si riferiscono ai 3 bus esterni 12c-like, mentre il 4300 al bus i2c interno.

Per scrivere, ad esempio, nei registri di I/O con indirizzo i2c 21(hex), si deve comunicare all'ELMB, tramite CANbus, che si vuole utilizzare l'oggetto che si trova all'indirizzo 4300 con sottoindirizzo 0, fornendogli come ingresso l'indirizzo 21 hex, poi con sottoindirizzo 1, inviando il numero di byte che si vuole scrivere, e così via secondo lo schema riportato in tabella 5.1.

Gli oggetti 4500 e 4501 permettono l'accensione e lo spegnimento dei quattro moduli ELMB. Richiamando l'oggetto con indirizzo 4500 e sottoindirizzo 1-4 si ottiene l'accensione (0xff) o lo spegnimento (0x00) del rispettivo modulo ELMB.

L'utilizzo di quest'oggetto è subordinato ad una precedente chiamata all'oggetto "Power Control Password" all'indirizzo 4501 dove è contenuta una chiave a 32 bit che va comunicata prima d'ogni operazione di accensione/spengimento in modo da rendere impossibile che una chiamata accidentale all'oggetto "Power Control" provochi uno spegnimento accidentale di uno o più moduli ELMB.

Questo evento è da evitare accuratamente, poiché lo spegnimento contemporaneo di tutti i moduli ELMB di una Service Board richiederebbe un power-up manuale della stessa.

Identico funzionamento è stato previsto per gli oggetti 4600 e 4601: il primo permette il reset delle schede FE connesse hai 12 canali LVDS della Service Board, mentre il secondo si occupa della protezione tramite password.

All'indirizzo CANOpen 4700 è implementato l'accesso ai registri dell'Actel. I 14 registri dell'actel sono indirizzati direttamente tramite il sottoindirizzo dell'oggetto e viene inviato dall'ELMB un messaggio d'errore se si tenta di scrivere su di un registro che non esiste. L'opera di conversione al giusto numero di bit del risultato dell'operazione di lettura nei registri dell'Actel, che per lo standard del protocollo CANOpen avviene a blocchi di 8 bit, è invece effettuata dal software client su PC.

Index (hex)	Sub Index	Name	Data/ Object	Attr	Default	Comment
		ELMB SB10 FV12				
4000		***I2C bus 0***	Record		Default	
	0	I2C ADDRESS	U8	RW	0xA0	
	1	I2C number of bytes	U8	RW	0xB0	
	2	I2C data bytes	U32	RW		
	3	I2C SubAddr	U8	RW	0xC0	
	4	I2C subaddr data bytes	U32	RW		
4100		***I2C bus 1***	Record		Default	
	0	I2C ADDRESS	U8	RW	0xA1	
	1	I2C number of bytes	U8	RW	0xB1	
	2	I2C data bytes	U32	RW		
	3	I2C SubAddr	U8	RW	0xC1	
	4	I2C subaddr data bytes	U32	RW		
4200		***I2C bus 2***	Record		Default	
	0	I2C ADDRESS	U8	RW	0xA2	
	1	I2C number of bytes	U8	RW	0xB2	
	2	I2C data bytes	U32	RW		
	3	I2C SubAddr	U8	RW	0xC2	
	4	I2C subaddr data bytes	U32	RW		
4300		***I2C bus3(G)***	Record		Default	
	0	I2C ADDRESS	U8	RW	0xA3	
	1	I2C number of bytes	U8	RW	0xB3	
	2	I2C data bytes	U32	RW		
	3	I2C SubAddr	U8	RW	0xC3	
	4	I2C subaddr data bytes	U32	RW		
4500		***ELMB_PWR_CTRL***	Record		Default	
	0	Power ELMB0	U8	RW	0x00	0x00 PWR OFF, 0xFF PWR ON
	1	Power ELMB1	U8	RW	0x00	0x00 PWR OFF, 0xFF PWR ON
	2	Power ELMB2	U8	RW	0x00	0x00 PWR OFF, 0xFF PWR ON
	3	Power ELMB3	U8	RW	0x00	0x00 PWR OFF, 0xFF PWR ON
4501		***ELMB_PWR_CTRL_PSW***	Record		Default	
	0	Power ELMB0	U32	RW	0x00000000	PWRC=0x50,0x57,0x52,0x43
4600		***ELMB_RST_CTRL***	Record		Default	
	0,1,..,11	Reset Longline Nr.	U8	RW	0x00	0x00 RESET OFF, 0xFF RESET ON
4601		***ELMB_RST_CTRL_PSW***	Record		Default	
	0	Power ELMB0	U32	RW	0x00000000	RSTC=0x52,0x53,0x54,0x43
4700		***Actel Registers***	Record		Default	
	0	*{6'h0,REG_TEST_PULSE_SEL [1:0]}*	U8	RW	0x00	
	1	*REG_TDIV [7:0]*	U8	RW	0x00	
	2	*{5'h0,REG_INT_PULSE_SEL[2:0]}*	U8	RW	0x00	
	3	*{7'h0,REG_PULSE };*	U8	RW	0x00	
	4	*REG_TMIS_L*	U8	RW	0x00	
	5	*REG_TMIS_H*	U8	RW	0x00	
	6	*REG_TDLY*	U8	RW	0x00	
5YXX		***Dialog Reg(i2c channel Y)addres XX***	Record		Default	Y=0,1,2 XX=0-7F
	0…15	PCH0…15: Delay + Shaper	U(5+3)	RW	0x00	
	16	*MSB PCH Mask (Ch15…Ch8)*	U8	RW	0x00	
	17	*LSB Mask (Ch7…Ch0)*	U8	RW	0x00	
	18	*Physical signals combination*	U3	RW	0x00	
	19	*DAC 0: Threshold ASD 0.*	U8	RW	0x00	
	20	*DAC 1: Threshold ASD 1.*	U8	RW	0x00	
	21	*DAC 2: DLL0 DAC*	U8	RW	0x00	
	22	*DAC 3: DLL1 DAC.*	U8	RW	0x00	
	23	*Test Mode configuration*	U4	RW	0x00	
	24	*ASD0 pulse mask + delay (1+5 bits)*	U6	RW	0x00	
	25	*ASD1 pulse mask + delay (1+5 bits)*	U6	RW	0x00	
	26	*Physical signal selection for Scaler*	U4	RW	0x00	
	27	MSB Rate counter	U8	RW	0x00	
	28	LSB Rate counter	U8	RW	0x00	
	29	Pattern generator Ch15…Ch8	U8	RW	0x00	
	30	Pattern generator Ch7…Ch0	U8	RW	0x00	

Tabella 5.1: Indirizzi CanOpen degli oggetti dell'SDO

Infine, nel range d'indirizzi CANOpen da 5000 a 52FF, sono riportate le funzioni d'accesso ai registri del DIALOG. Siccome ogni canale i2c può servire un "long i2c branches" con 10 DIALOG chip, quindi ogni ELMB può gestire 30 DIALOG sullo stesso ramo con indirizzo diverso, è stato creato un oggetto generico 5YXX in cui Y rappresenta il canale LVDS d'interesse (0-2) e XX l'indirizzo i2c del chip a cui si vuole accedere. Questo fornisce un accesso ai registri efficiente e di facile lettura, rimandando al software client la gestione delle funzionalità del DIALOG ancora in fase di test. Se, per esempio, si vuole accedere ad uno dei registri di un DIALOG posto sul canale 1 con indirizzo 33, si dovrà utilizzare l'oggetto all'indirizzo 5133 con sottoindirizzo uguale al registro a cui si vuole accedere.

	2AA00 EEPROM	
I2C ADDRESS	**I2C Subaddress**	**Description**
50h-57h	00h-010h	Read-only registers

	PIO0 PCF8575	
I2C ADDRESS	**I2C DATA**	**Description**
20h	Data port 0 , Data port 1	

	PIO1 PCF8575	
I2C ADDRESS	**I2C DATA**	**Description**
21h	Data port 0 , Data port 1	

Actel FPGA		
I2C ADDRESS	**I2C Subaddress**	**Description**
72h		
	00h	***{6'h0,REG_TEST_PULSE_SEL [1:0]}***
	01h	***REG_TDIV [7:0]***
	02h	***{5'h0,REG_INT_PULSE_SEL[2:0]}***
	03h	***{7'h0,REG_PULSE };***
	04h	***REG_TMIS_L***
	05h	***REG_TMIS_H***
	06h	***REG_TDLY***

Tabella 5.2: Mappa degli indirizzi i2c.

5.3 Il software client e il test.

Il protocollo CANOpen e il CAN-bus permettono il controllo delle Service Board da PC remoto tramite le schede Kvaser PCI-CAN. A questo scopo è stato sviluppato un software-client in linguaggio c, compilato con compilatore "Microsoft-Visual C++", che offre un rapido accesso alle caratteristiche e alle funzionalità della Service Board tramite dei menu di navigazione testuali.

Durante il lavoro di tesi è stata scritta parte di tale software, in particolare quella necessaria alla gestione dei nuovi oggetti CANOpen implementati nel firmware, precedentemente descritti. Inoltre è stato eseguito il test dell'hardware controllato da questo software e del chip DIALOG, sopratutto in relazione al suo utilizzo da parte della Service Board.

Nei paragrafi successivi sarà descritta l'interfaccia utente con i suoi menù e il risultato dei test.

5.3.1 Menu Principale

In figura 5.12 è riportata la schermata del menù principale del software client.

Selezionando una delle lettere visualizzate si accende ad una serie di sottomenu che permettono il monitoraggio e il controllo della Service Board, delle sue funzionalità e delle periferiche ad essa collegate (schede di FE).

In fase iniziale è importante che la variabile NodeID corrisponda a quella dell'ELMB con cui si vuole interagire. Il suo valore è mostrato infondo alla schermata. NodeID può essere impostato selezionando "n change node number" da questo menu.

Sempre in fase di inizializzazione è opportuno effettuare il reset dell'apparato. Selezionando **r** viene inviato il messaggio di reset sul CAN e tutte le ELMB vengono riavviate. A seguito di quest'operazione il software-client riceverà un messaggio di avvio del sistema da parte d'ogni ELMB riavviata.

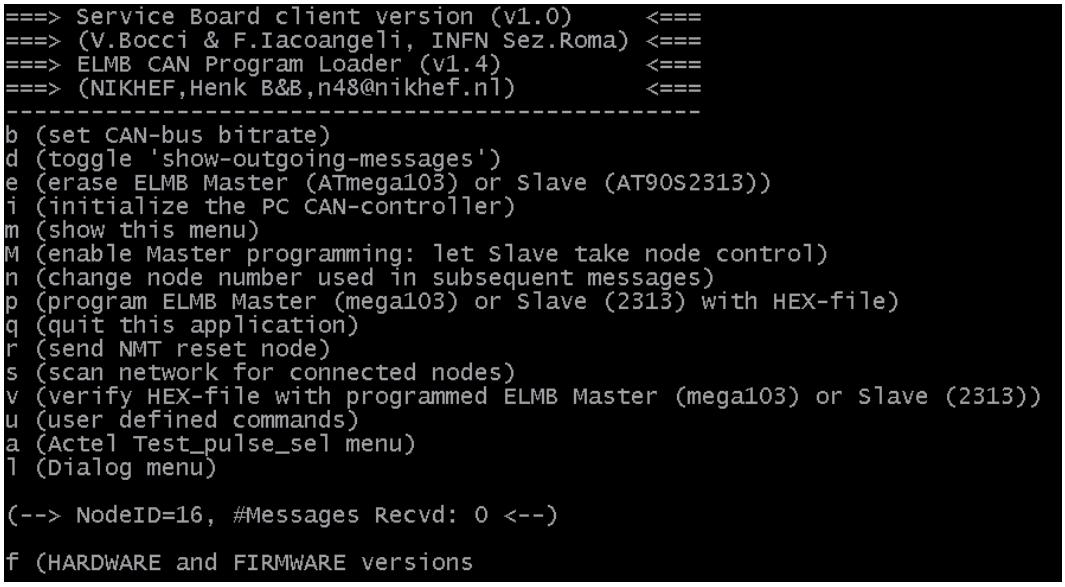


Figura 5.12: Menù principale del software client.

Per controllare quali ELMB sono disponibili sul nodo CAN e a quali sottoindirizzi si può effettuare la scansione delle periferiche collegate. In questo caso si riceverà un messaggio di risposta da ogni ELMB interrogata contenente lo stato, il sottoindirizzo i2c e la versione del firmware. La versione è anche visualizzabile selezionando **v** dal menu principale.

5.3.2 Actel Menu

L'Actel menu rispecchia fedelmente la struttura interna dell'Actel e dei suoi registri ed è stato strutturato seguendo quest'idea: ogni volta che si accede al menu si fa un'operazione di lettura dei due registri principali dell'Actel, cioè quelli che ne definiscono le linee abilitate nei due multiplexer (figura 5.9), e un'apposita funzione maschera le funzionalità disponibili e il menù testuale, in modo da presentare le giuste opzioni. In ogni schermata è presente una struttura ad albero che mostra lo stato attuale dei registri interni: così, ogni volta che si ha necessità di modificare i registri dell'Actel, si potrà accedere al menu avendo subito un'idea chiara dei valori ivi contenuti.
Il digramma di flusso di questo menu è riportato in figura 5.13.

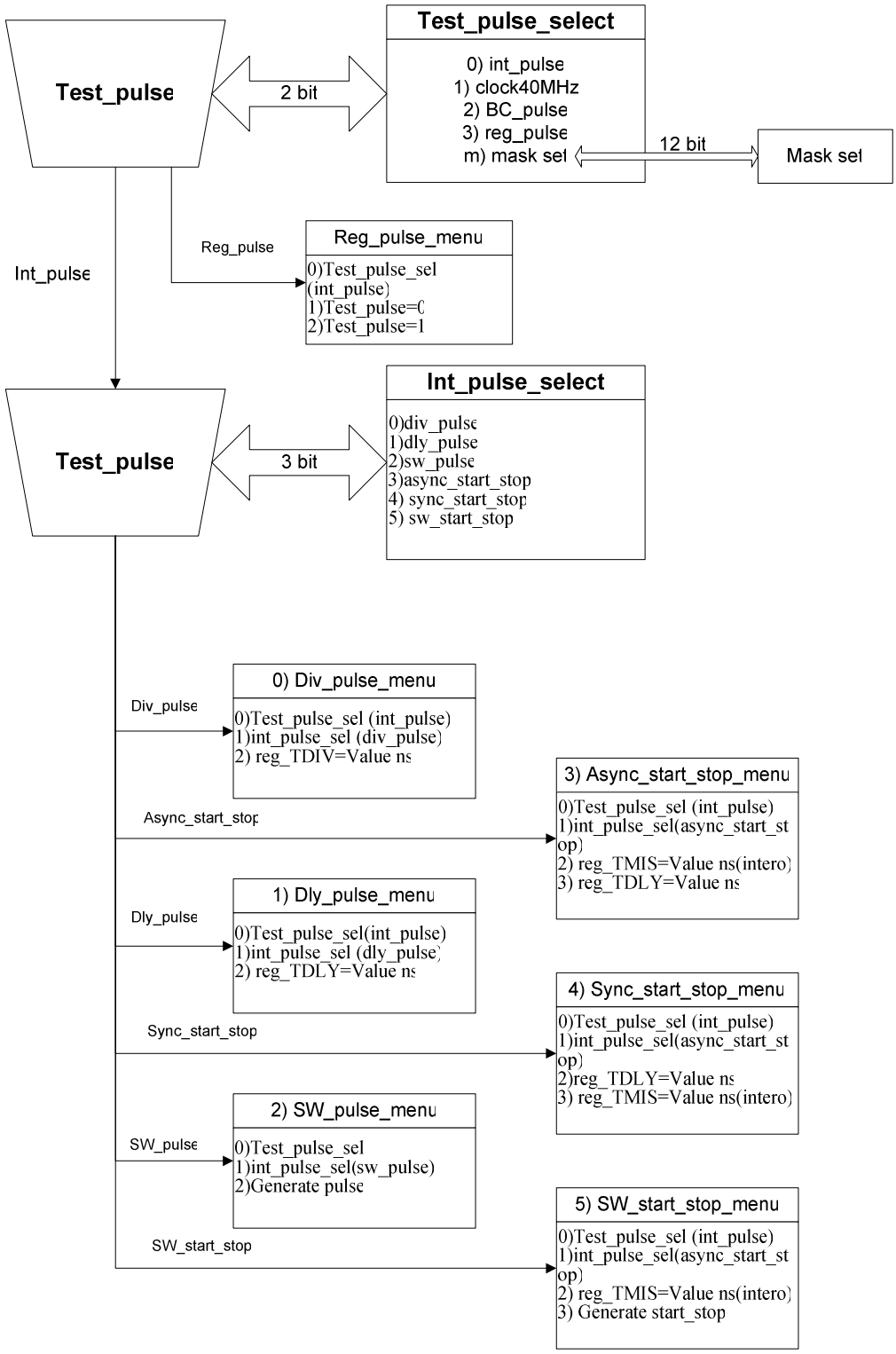


Figura 5.13: Diagramma di flusso del menu per la gestione delle funzioni dell'Actel (Actel_menu)

La funzione principale dell'FPGA Actel è quella di fornire in uscita un certo numero di segnali impulsivi che permettono il test e il controllo dell'eletronica di front-end. Impostando nel giusto modo i suoi registri è possibile selezionare tali segnali e le loro caratteristiche che verranno inviate all'ELMB lungo una linea denominata "test_pulse". L'ELMB renderà infine disponibile questa linea sui canali i2c-like.
Per verificare il giusto funzionamento dell'Actel e del menu di gestione è stato effettuato il test di tutti i possibili segnali di uscita, osservandoli con l'oscilloscopio. Nel paragrafo successivo sono riportati i risultati dei test e, contemporaneamente, è descritta la navigazione nel menu.

- Funzionamento e test

Seguendo il diagramma di flusso riportato in figura 5.13 la prima schermata che si presenta, ad un primo accesso all'Actel, permette di configurare il registro Test_pulse.

Figura 5.14: Menu di configurazione del registro Test_pulse dell'Actel

.

Come si vede è possibile abilitare 4 canali diversi nel multiplexer a cui si riferisce. Osserviamo che qui, come in quasi tutti i menu relativi all'Actel, il numero corrispondente alla scelta del canale da abilitare corrisponde al valore effettivamente scritto all'interno del "Tst_Pulse_reg".
Se si seleziona "clk40 set" si rende disponibile direttamente il clock a 40 MHz di LHC ottenendo in uscita su test_pulse il segnale mostrato nella figura seguente.

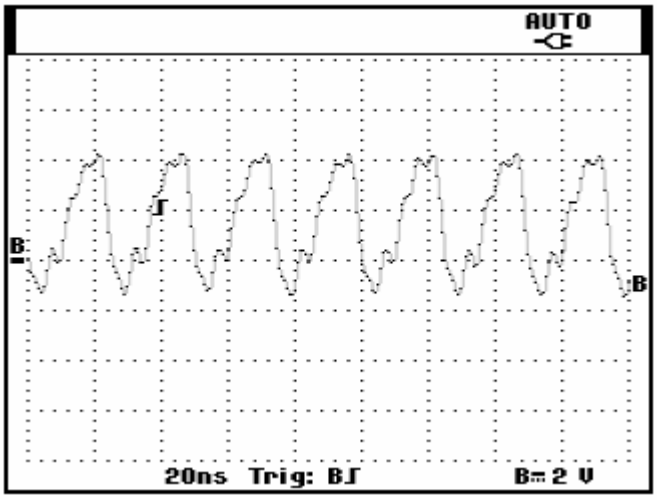


Figura 5.15: Il clock a 40MHz osservato sulla linea test_pulse selezionandolo attraverso i multiplexer dell'Actel.

Il disturbo che si osserva nel segnale, che dovrebbe essere ad onda quadra, è causato all'impossibilità di riferirlo alla giusta massa.
Si noti inoltre che, da questo menu come da ogni altro dell'Actel, è possibile eseguire una lettura o una scrittura diretta dei registri, in modo da semplificare eventuali operazioni di debugging.

```
**************************************************************
                clk40  _menu
**************************************************************
t (test_pulse_source) [clk40  ]                              *
x read/write reg                                             *
**************************************************************
```

Figura 5.16: Il clk40_menu

Impostando a 2 il registro test_pulse_reg si abilita un segnale impulsivo proveniente dall'esterno, denominato BC_pulse, che nel rivelatore di LHCb sarà fornito dal modulo PDM. Il test è stato eseguito simulando il segnale con un impulso quadrato d'altezza ~3V e durata 80ns, ripetuto con una periodicità di 300μs.

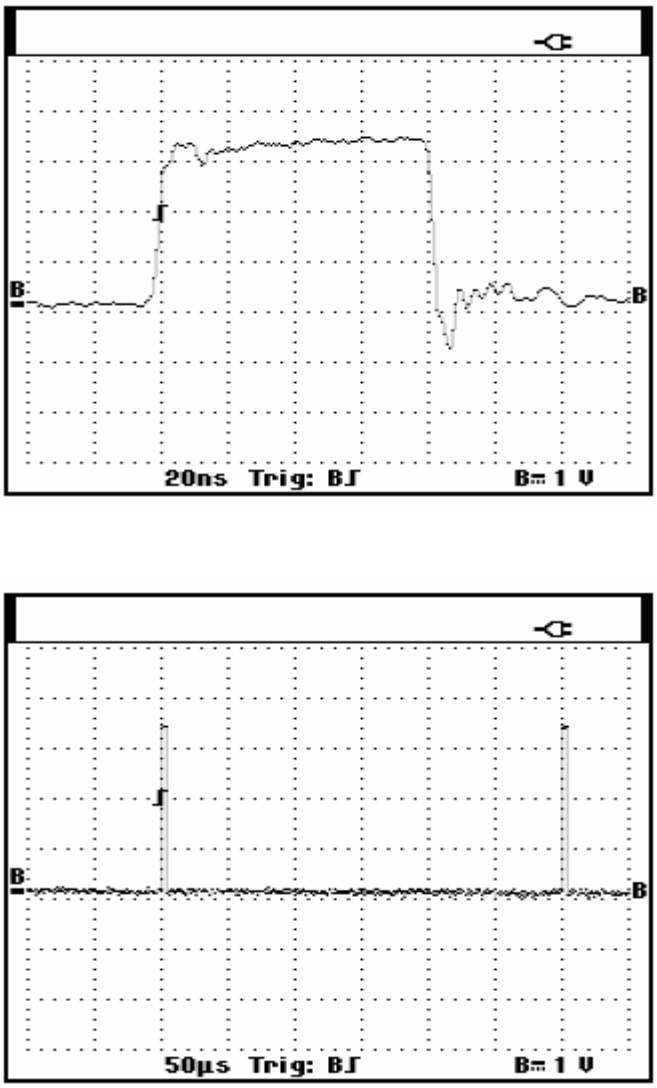


Figura 5.17: Il BC_pulse all'oscilloscopio osservato su scale diverse per testarne sia la forma (a) che la periodicità (b).

Il BC_pulse rende possibile, ad esempio, eseguire una misura ad un istante pilotato da un segnale esterno di forma e durata qualsivoglia, nel range coperto dalle caratteristiche del sistema.

Figura 5.18: Il BC_pulse_menu

Se, dal Test_pulse_select, si seleziona "reg_pulse_menu", si accede ad un sottomenù che permette di portare la linea d'uscita alta o bassa, oppure di fare un "toggle", scrivendo nel registro ad 1 bit Reg_pulse.

Figura 5.19: Il reg_pulse_menu

Come si può vedere in figura 5.19, è riportato sia lo stato del registro Test_pulse (prima riga tra parentesi quadra), sia quello del registro che è possibile modificare (penultima riga).
Un toggle sul registro ha l'effetto di negare il suo stato, producendo sulla linea test_pulse un cambiamento di stato da basso ad alto o viceversa.
Anche questa funzione è stata testata studiando l'uscita con l'oscilloscopio e ottenendo il risultato in figura 5.20.

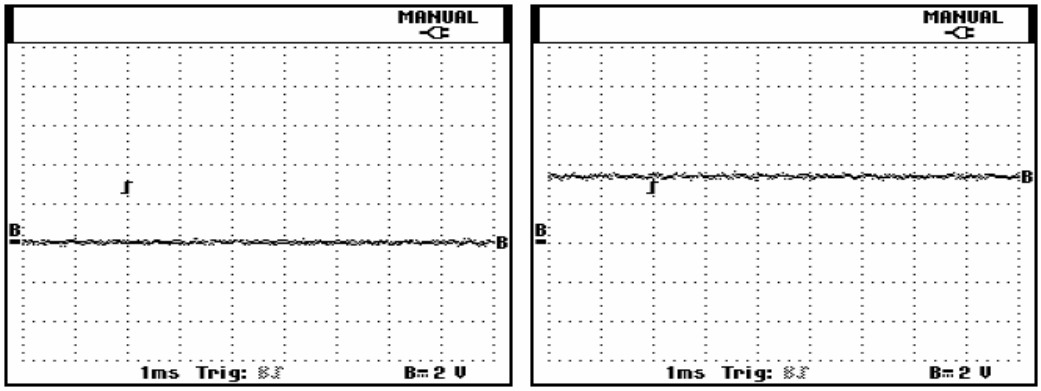


Figura 5.20: Toggle 0→1 nel registro reg_pulse.

Come in tutti i sottomenu, è possibile tornare al menu di configurazione del multiplexer test_pulse digitando t.

Infine, selezionando il canale 0 nel test_pulse_select, s'abilità l'uscita di un secondo multiplexer denominato int_pulse, come mostrato nello schema a blocchi dell'actel (fig. 5.9).

È inoltre mostrata, all'interno della schermata Test_Pulse_select, una tabella che riporta lo stato dei bit di maschera dei dodici canali test_pulse, modificabile selezionando m.

.

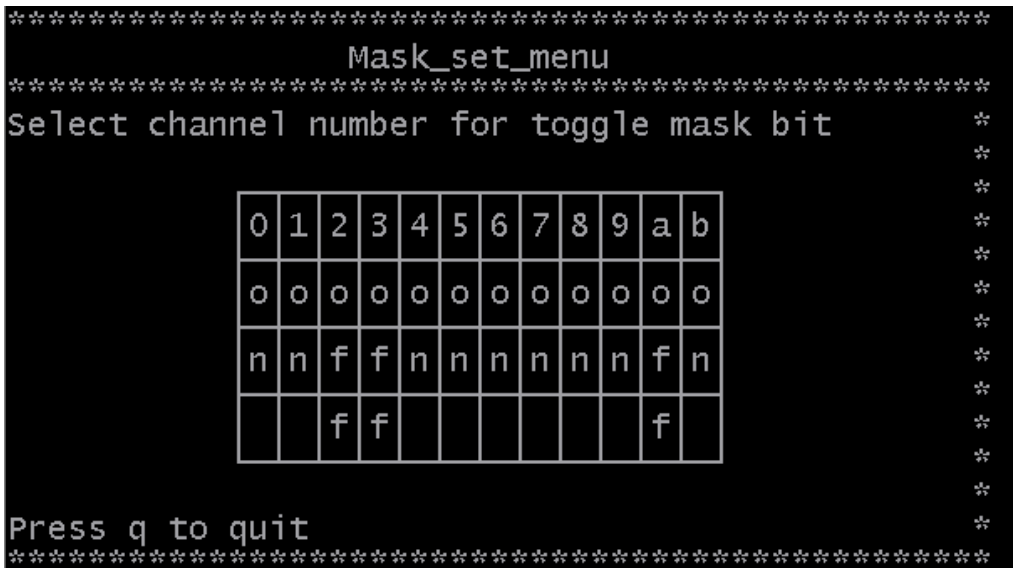


Figura 5.21: Mask_set_menu

In questo caso non si effettua una scrittura nei registri, ma si accede ad un successivo menu da cui è possibile eseguire un'operazione di toggle sul valore d'ogni singolo bit di maschera digitando il tasto corrispondente.
Anche qui una tabella autoaggiornante riporta lo stato dei canali.
Torniamo ora al secondo multiplexer, int_pulse, al quale fa riferimento un altro menu di selezione dei canali, denominato int_pulse-select.

Figura 5.22: Int_pulse_select

Al valore selezionato corrisponde quello realmente scritto all'interno del registro int_pulse. Questo menu è molto importante perché permette di selezionare un gran numero di segnali utili per il test ed il controllo dell'elettronica di front-end.
Selezionando "div_pulse menu" viene generato un segnale impulsivo ad onda quadra di durata 25ns (cioè uguale al clock) con periodo, multiplo intero di quello di clock, riportato nel registro div_pulse a 8bit. Quando ci si trova in questo stato del multiplexer viene visualizzato il sottomenu div_pulse_menu.

Figura 5.23: div_pulse_menu

Nel menu è visualizzata, come sempre, la struttura ad albero dei registri e il valore in ns del periodo del segnale, con il relativo fattore moltiplicativo del clock tra parentesi, modificabile scrivendo nel registro reg_TDIV.
Lo studio dell'uscita con l'oscilloscopio del corretto funzionamento di questa caratteristica ha dato esito positivo fornendo i dati in figura 5.24a/b.

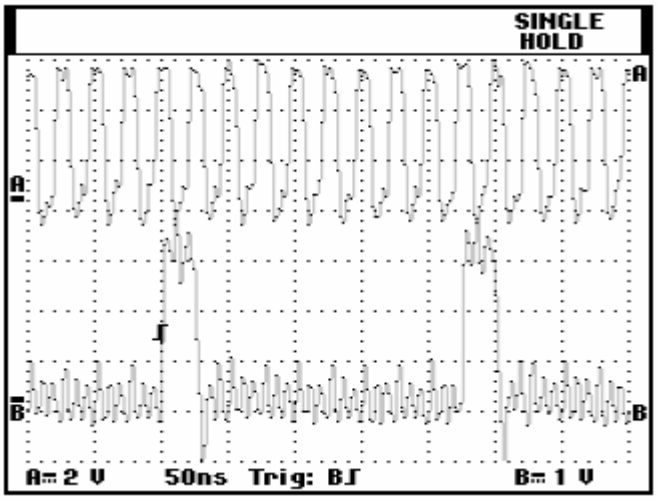


Figura 5.24a: test della linea div_pulse del multiplexer con reg_TDIV=8 (200ns). Sul canale A dell'oscilloscopio è possibile osservare il clock 40 per verificare la periodicità del segnale.

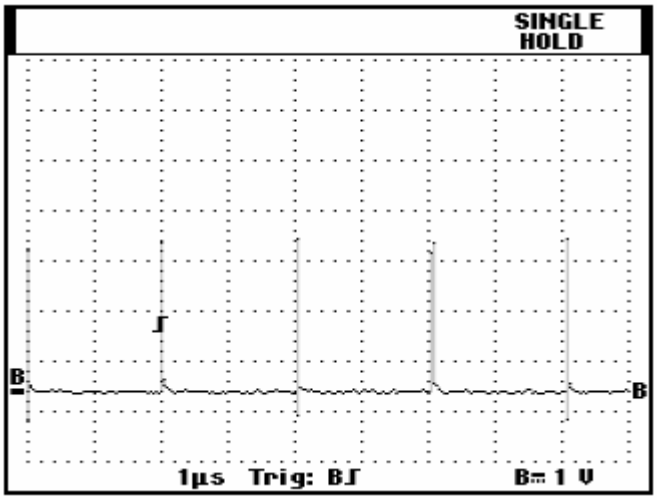


Figura 5.24b: test della linea div_pulse del multiplexer con reg_TDIV=80 (2000ns).

Essendo il registro in questione a 8bit, è possibile impostare un periodo, per l'impulso prodotto, compreso tra 0 e 255*25=6375 ns a passi di 25 ns.
Se si seleziona "dly_pulse menu" si utilizza il segnale BC_pulse come riferimento per generare un segnale impulsivo, della forma di un colpo di clock, con ritardo variabile.

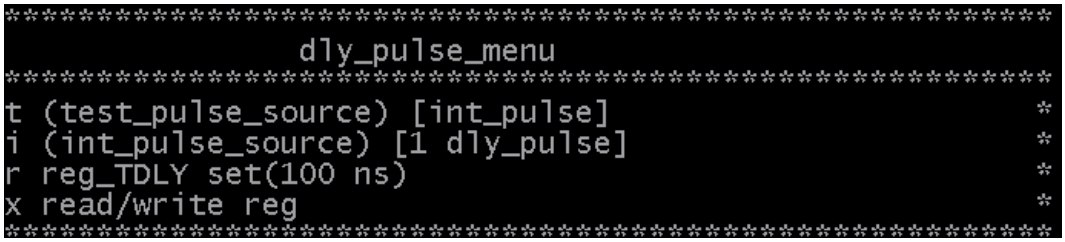


Figura 5.25: dly_pulse_menu

È possibile impostare la durata del ritardo, scrivendone il valore, come multiplo del periodo di clock, nel registro reg_TDLY, anch'esso di 8 bit come reg_DIV e quindi con le stesse caratteristiche in quanto a range e passo.
Questo segnale può servire, ad esempio, a fornire un segnale di start o di stop dopo un tempo prefissato da un certo evento. In figura 5.26 sono riportati il segnale BC_pulse e quello ritardato del tempo voluto studiati all'oscilloscopio, utilizzando come segnale di test un impulso esterno di 100 ns.

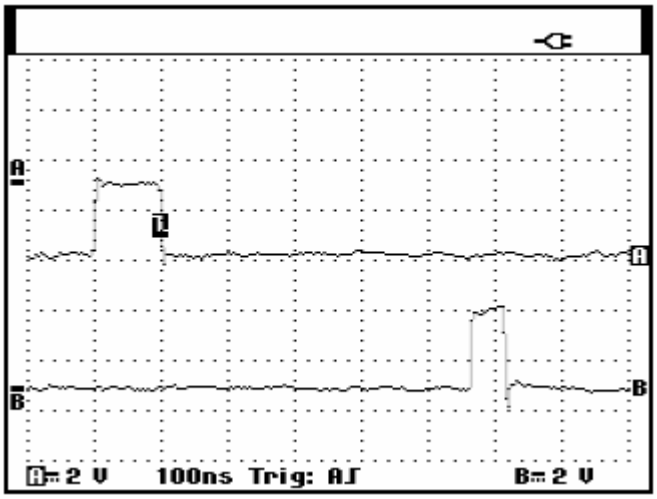


Figura 5.26: il segnale prodotto selezionando dly_puse nel multiplexer con il ritardo impostato a 500ns.

Anche da tale menu, come da tutti quelli selezionabile in questo livello del programma, è possibile tornare ai menu di modifica dei registri d'entrambi i multiplexer.

Selezionando nel multiplexer la linea "sw_pulse" (software pulse) è possibile generare a comando un segnale impulsivo quadrato, della durata di un colpo di clock, come indicato dalla grafica corrispondente (fig. 5.27).

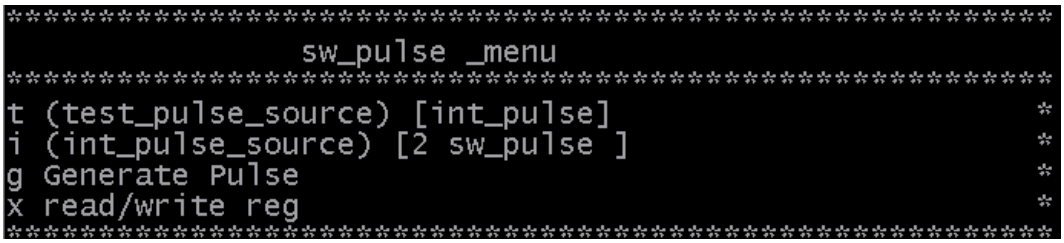


Figura 5.27: sw_pulse_menu

Anche qui lo studio all'oscilloscopio ha generato i risultati riportati in figura .

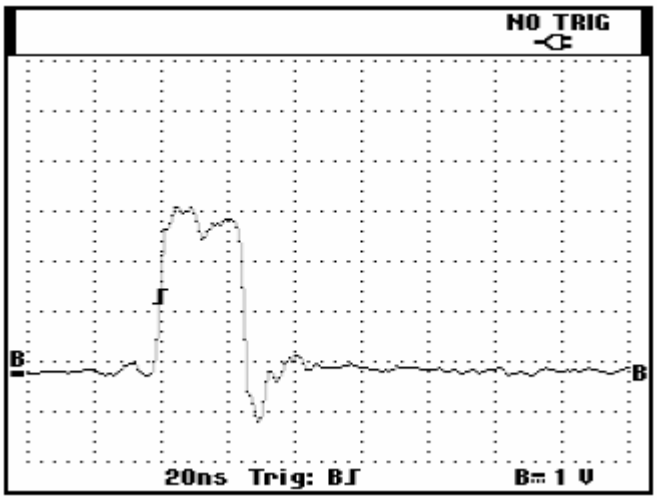


Figura 5.28: Impulso generato dalla funzione sw_pulse dell'Actel.

Infine le ultime 3 selezioni possibili dal menu del multiplexer int_pulse, corrispondenti ai valori 3,4 e 5, permettono di generare un segnale di start/stop per l'elettronica di front-end comandato dagli impulsi visti in precedenza.

Perché venga effettuato uno start/stop è necessario presentare due segnali impulsivi alla distanza voluta, di cui il primo viene interpretato come start e il successivo come stop. Il tempo che intercorre tra i due impulsi è la durata della misura stessa che l'elettronica di front-end deve eseguire a seguito dello start. questo tempo verrà chiamato nel seguito TMIS, in analogia col nome del registro che lo controlla. Il registro reg_TMIS dispone di 16 bit e permette quindi di gestire un range compreso tra 0 e 65536*25=1638400 ns. La parte alta e la parte bassa del registro sono indirizzate singolarmente, per rendere possibile l'accesso tramite i2c-bus.

"Async_start_stop" permette di eseguire l'operazione start/stop utilizzando come riferimento il segnale prodotto allo stesso modo che nella funzione div_pulse.

Figura 5.29: Async_start_stop_menu

Dal menu corrispondente è possibile selezionare il tempo di misura TMIS e la larghezza tra i due impulsi TDIV della linea div_pulse.

Lo stesso tipo di segnale può essere pilotato dal segnale prodotto dalla funzione dly_pulse selezionando "Sync_start_stop".

Infine è possibile effettuare uno start_stop software pilotato da un impulso generato manualmente, selezionando il canale "sw_start_stop" del multiplexer. In questo caso, alla pressione del tasto (g), si otterrà la generazione dei due impulsi di start e di stop, distanziati tra loro del tempo impostato in TMIS.

```
*****************************************************************
                sync_start_stop_menu
*****************************************************************
t (test_pulse_source) [int_pulse]                               *
i (int_pulse_source) [5 sync_start_stop]                        *
r reg_TDLY set(100 ns)                                          *
m reg_TMIS set(10000 ns)                                        *
x read/write reg                                                *
*****************************************************************
```

Figura 5.30: Sync_start_stop_menu

```
*****************************************************************
                sw_start_stop_menu
*****************************************************************
t (test_pulse_source) [int_pulse]                               *
i (int_pulse_source) [6 sw_start_stop]                          *
m reg_TMIS set(10000 ns)                                        *
g Generate Pulse                                                *
x read/write reg                                                *
*****************************************************************
```

Figura 5.31: sw_start_stop_menu

Anche queste caratteristiche sono state studiate osservandone il risultato lungo la linea di test_pulse e ottenendo all'oscilloscopio, i risultati mostrati nelle figure seguente.

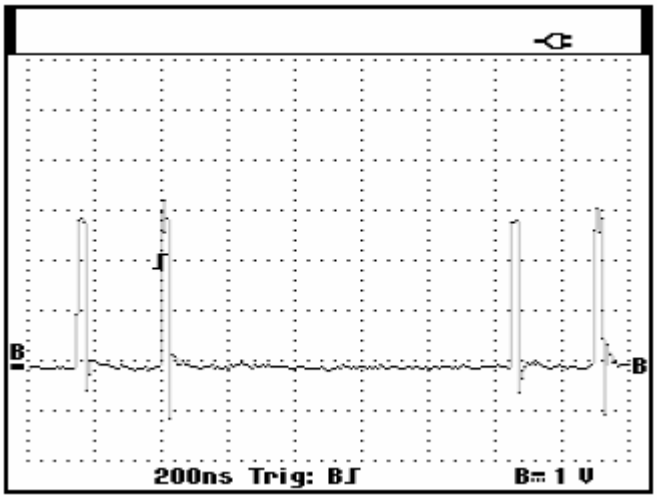


Figura 5.32: segnali di start e stop generati selezionando Async_start_stop. Lo start e stop sono composti di due segnali prodotti dalla funzione div_pulse.

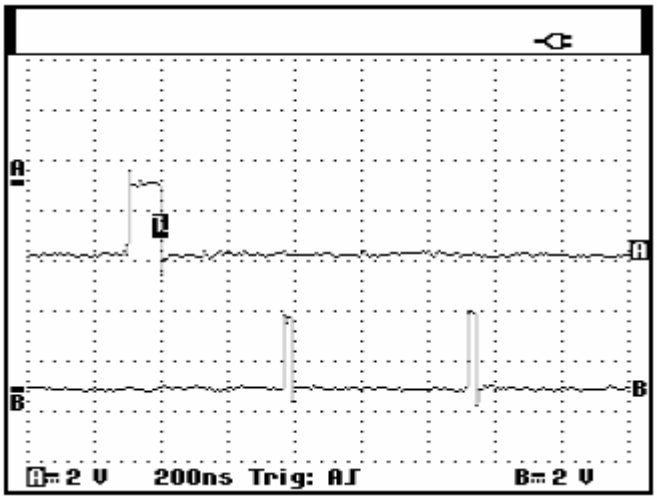


Figura 5.33: Sync_start_stop. Nel canale A dell'oscilloscopio è mostrato il segnale BC_pulse, mentre nel canale B il segnale prodotto.

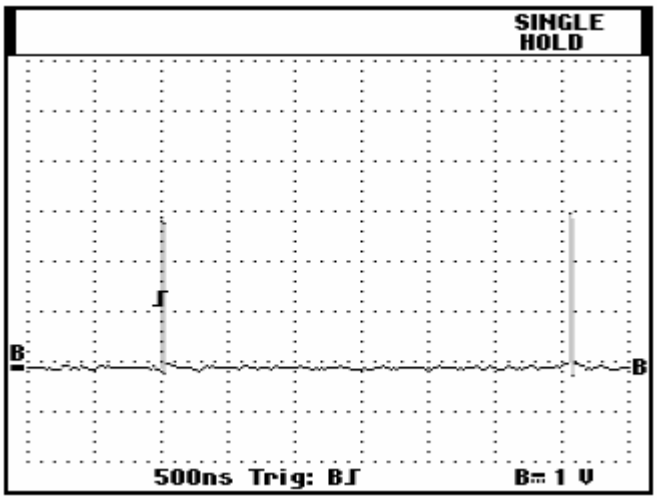


Figura 5.34: start_stop software realizzato utilizzando un TMIS di 80 colpi di clock corrispondenti ad un tempo di circa 3μs.

5.3.3 User Menu

Da questo menu è possibile utilizzare gli oggetti 4500 e 4600 implementati nel firmware del modulo ELMB, ovvero le funzioni di "power cycling" dei 4 moduli e di "reset" delle 12 linee LVDS della Service Board. Queste operazioni sono effettuate dal software client eseguendo le procedure descritte nel paragrafo dedicato all'SDO del firmware. Ogni procedura viene preceduta dall'invio della giusta password (in fase di debugging impostata come 52.53.54.43) rispettivamente all'indirizzo 4501 e 4601.

Figura 5.35: User_menu

Se si vuole effettuare il reset di una delle longline LVDS si accede ad una schermata di scelta da dove è possibile eseguire un qualunque numero di reset selezionando, ad una ad una, le linee d'interesse.

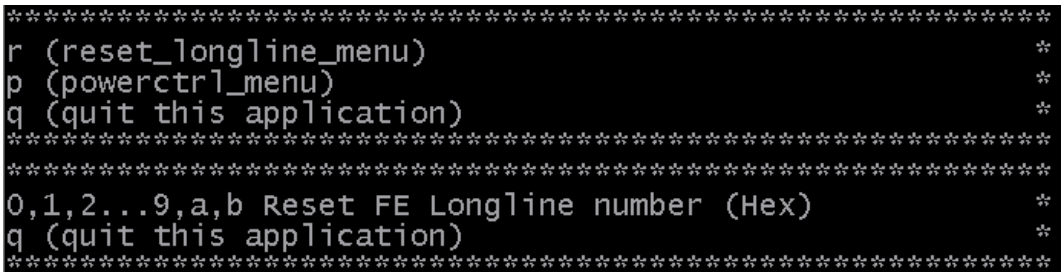


Figura 5.36: schermata di selezione della linea su cui effettua il reset.

A seguito d'ogni selezione si riceve un messaggio d'avvenuto reset della linea o, se la scelta non corrisponde a nessuna delle linee, un messaggio d'errore.
In particolare la numerazione 0,1,..9,a,b si riferisce ad un ordinamento consecutivo delle linee partendo dalla più lontana dalla porta d'ingresso del nodo can, come mostrato in figura 5.37.

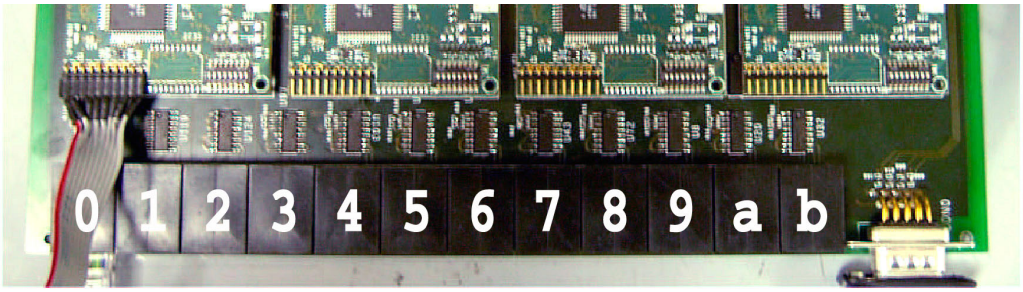


Figura 5.38: Connettori per FE-longline-LVDS.

Il reset avviene scrivendo prima 1 e poi 0 nel bit relativo alla linea scelta, contenuto in uno dei due registri di I/O all'indirizzo i2c 0x20.

Per effettuare un power cycling di un modulo ELMB si procede in modo del tutto analogo. Anche stavolta selezionando il numero del canale interno relativo al modulo su cui si vuole agire si ottiene un messaggio d'operazione avvenuta, altrimenti si riceve un messaggio d'errore.

```
**************************************************************
r (reset_longline_menu)                                      *
p (powerctrl_menu)                                           *
q (quit this application)                                    *
**************************************************************
**************************************************************
0,1,2,3 Power Cycle ELMB number (Hex)                        *
q (quit this application)                                    *
**************************************************************
```

Figura 5.39: schermata di selezione dell'ELMB su cui il power cycling.

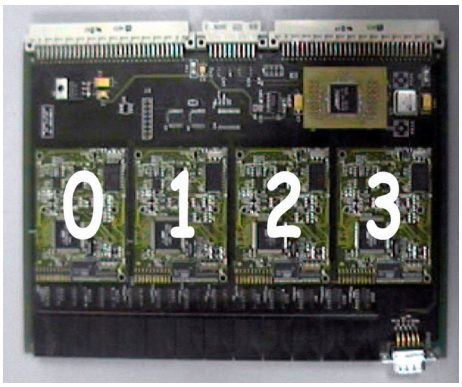


Figura 5.38: Numerazioni interna delle ELMB

Allo stesso modo il canale 0 si riferisce alla prima ELMB a sinistra rispetto al connettore del nodo CAN. Come per i connettori delle linee LVDS, la numerazione rispecchia quella dei canali interni della Service Board ed è indipendente dall'indirizzo i2c del modulo.

L'operazione di power cycling consiste nel portare sotto il livello minimo di funzionamento la tensione d'alimentazione dell'ELMB per un certo tempo e poi riportarla alle condizioni di funzionamento provocandone il riavvio.
Ciò avviene in maniera analoga al reset: scrivendo uno nel giusto registro di I/O e riscrivendo 0 dopo un tempo abbastanza lungo da poter trascurare tutti gli effetti capacitivi.
A seguito di quest'operazione viene inviato al nodo CAN un messaggio d'avvenuto riavvio dell'ELMB che assicura il corretto funzionamento di questa caratteristica. Il suo funzionamento è stato testato su tutti i 4 canali interni delle ELMB.

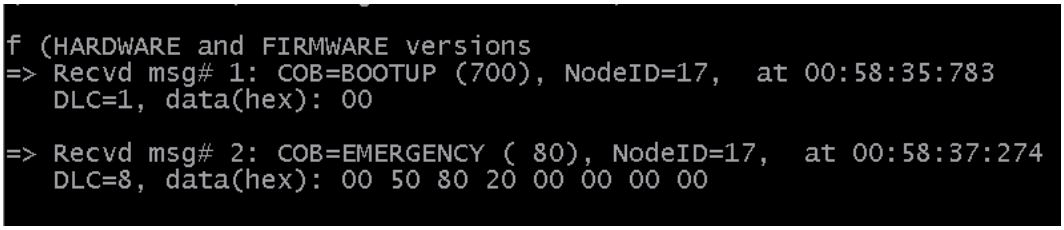


Figura 5.40: Messaggio d'avvenuto riavvio a seguito di un'operazione di power cycling.

Quando si utilizza questa funzione è importante evitare di effettuare un power cycling sul modulo che si sta utilizzando. Se così avviene l'ELMB effettuerà regolarmente il primo accesso ai registri di I/O, spegnendosi, ma, non potendo effettuare il secondo accesso, rimarrà in questo stato. Al verificarsi di quest'eventualità si riceveranno sul nodo CAN una serie di messaggi di timeout.
Per ravviare l'ELMB spento bisognerà operare un power cycling manualmente, agendo direttamente sull'alimentazione, o, in alternativa, effettuare la seconda scrittura sui registri tramite un'altra ELMB.

5.3.4 DIALOG Test Menu

Questo menu e le funzioni accessibili sono stati pensati per permettere un rapido test dei chip DIALOG controllati dalla Service Board.

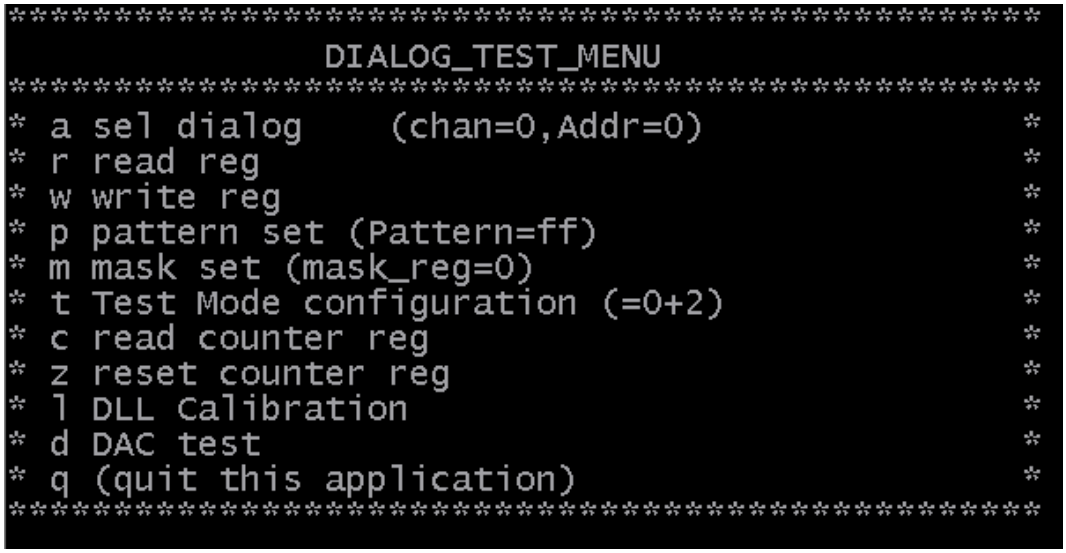


Figura 5.41: Dialog_test_menu.

Selezionando “sel dialog” si può modificare l’indirizzo i2c del DIALOG con cui si vuole interagire, permettendo così di comunicare con più di un chip su ogni braccio LVDS, e il canale i2c su cui è connesso. Tra parentesi sono riportati i valori correnti di tali parametri.

È anche possibile eseguire una scrittura o una lettura diretta sui 31 registri del chip DIALOG permettendo così di accedere manualmente a tutte le su funzionalità.

L’indirizzamento ai registri va effettuato dando al programma client il suo indirizzo decimale: tale scelta è stata fatta per mantenere lo stesso formato riportato nel data sheet del DIALOG chip. [Ref.20] Il valore dei registri invece si intende esadecimale.

In particolare in fase di lettura dei registri viene restituito il valore nel formato indicato in tabella 5.3.

Locatio	Bits	Comments
0	5 + 3	PCH0: Delay + Shaper
1	5 + 3	PCH1: Delay + Shaper
2	5 + 3	PCH2 Delay + Shaper
3	5 + 3	PCH3: Delay + Shaper
4	5 + 3	PCH4: Delay + Shaper
5	5 + 3	PCH5: Delay + Shaper
6	5 + 3	PCH6: Delay + Shaper
7	5 + 3	PCH7: Delay + Shaper
8	5 + 3	PCH8: Delay + Shaper
9	5 + 3	PCH9: Delay + Shaper
10	5 + 3	PCH10: Delay + Shaper
11	5 + 3	PCH11: Delay + Shaper
12	5 + 3	PCH12: Delay + Shaper
13	5 + 3	PCH13: Delay + Shaper
14	5 + 3	PCH14: Delay + Shaper
15	5 + 3	PCH15: Delay + Shaper
16	8	MSB PCH Mask (Ch15…Ch8)
17	8	LSB PCH Mask (Ch7…Ch0)
18	3	Physical signals combination
19	8	DAC 0: Threshold ASD 0.
20	8	DAC 1: Threshold ASD 1.
21	8	DAC 2: DLL0 DAC.
22	8	DAC 3: DLL1 DAC.
23	4	Test Mode configuration
24	6	ASD0 pulse mask + delay (1+5 bits)
25	6	ASD1 pulse mask + delay (1+5 bits)
26	4	Physical signal selection for Scaler
27	8	MSB Rate counter
28	8	LSB Rate counter
29	8	Pattern generator Ch15…Ch8
30	8	Pattern generator Ch7…Ch0

Tabella 5.3: sottoindirizzi i2c dei registri del DIALOG.

- Utilizzo e Test

Per velocizzare e semplificare il test sono state inserite funzioni di modifica dei registri di maschera (16 e 17), dei registri di “pattern generator” (29 e 30) e del registro “Test mode configuration” (23) accessibili direttamente dalla schermata principale del “DIALOG Test Menu”. Il valore di tali registri è riportato tra parentesi nella stringa di selezione delle funzioni d’accesso, in modo da visionarne lo stato ad ogni accesso al DIALOG.

Se si vuole modificare la maschera dei canali d'ingresso o il pattern, verrà chiesto di inserire il valore, in esadecimale, da assegnare a questi registri a 16bit dopodichè verrà effettuata la scrittura nel DIALOG utilizzando il corretto oggetto CAN.

Questi registri sono di cruciale importanza nell'utilizzo del DIALOG poiché stabiliscono condizioni sul suo ingresso: i bit di maschera permettono di abilitare o meno i canali fisici uno ad uno, in modo da poter individuare canali non funzionanti, mentre i bit contenuti nel pattern generator vengono utilizzati come ingresso, al posto dei canali fisici, quando richiesto.

Selezionando "Test mode configuration" si accede invece ad un sottomenu dove è possibile configurare il relativo registro.

Figura 5.42: sottomenu di configurazione del registro "Test Mode" del DIALOG.

Prima di andare avanti nella descrizione delle funzionalità del DIALOG bisogna mettere in luce il ruolo fondamentale che ha, per il funzionamento del chip, la ricezione del segnale test_pulse dalla Service Board.

Tale segnale, generato dall'Actel, viene utilizzato come riferimento per molte funzionalità del chip DIALOG. Esso può fornire il clock di riferimento per la calibrazione del DLL, o il segnale di start/stop per il contatore a 16bit, oppure comandare l'invio dei bit contenuti nel pattern generator o, ancora, generare un "ASD pulse".

Modificando il valore del registro "Test Mode Configuration" si configura un multiplexer che permette di instradare il segnale di test_pulse verso i suoi diversi utilizzi, secondo lo schema riportato in tabella 5.4.

Test Mode configuration	Commenti
X000	Disabilitato : non è possibile nessuna operazione con il segnale test_pulse
X001	ASD pulse mode : manda il segnale test_pulse alle schede ASD
X010	Start/Stop del Rate counter
X011	Patter generator : Abilita l'invio del Pattern contenuto nei registri 29 e 30 se test_pulse posedge.
X100	Calibration DLL : Abilita l'utilizzo del segnale test_pulse come clock di riferimento per la calibrazione del DLL
X101	Disabilitato : non è possibile nessuna operazione con il segnale test_pulse
X110	Disabilitato : non è possibile nessuna operazione con il segnale test_pulse
X111	Disabilitato : non è possibile nessuna operazione con il segnale test_pulse
1XXX	Abilita il refreshing dei registri di configurazione

Tabella 5.4: modalità selezionabili agendo sul registro Test Mode Configuration e relativo valore dei suoi quattro bit.

Il "Test Mode Configuration" non va pensato solo come registro di configurazione del multiplexer: infatti, oltre hai primi tre bit che riguardano esclusivamente il multiplexer della linea di test_pulse, è presente un quarto bit che permette di abilitare un sistema automatico di rivelamento errori e di aggiornamento dei registri, che lo protegge contro errori di tipo SEU.

Per maggiori approfondimenti a riguardo si faccia riferimento al data-sheet del DIALOG. [REF 20]

Le selezioni 0-7 riguardano i primi tre bit del registro, lasciando immutato il più significativo, mentre la selezione 8 effettua un toggle del bit più significativo. Si tenga presente che, per quanto riguarda le selezioni 0-7, il numero digitato corrisponde a quello effettivamente scritto nei 3 bit meno significativi del registro. Con 0,5,6 e 7 non si abilita nessun canale all'interno del multiplexer e il segnale di test_pulse viene ignorato dal chip.

"ASD pulse mode" fa si che il segnale venga utilizzato per generare due segnali impulsivi, ogni volta che la linea di test_pulse ha una transizione positiva _|¯ , che

vengono inviati ai due ASD eventualmente collegati al DIALOG. Le due uscite, “AD0_pulse” e “ASD1_pulse”, possono essere mascherate e ritardate singolarmente agendo rispettivamente sui registri a 6 bit 24 e 25. In particolare il bit più significativo contiene lo stato dell’uscita (1=enable/0=disable) e i rimanenti 5 bit il ritardato in 32 passi da 1,5ns.

Impostando il multiplexer su “Rate counter Start/Stop” si utilizzano le transizioni positive _|¯ della linea test_pulse per cambiare lo stato del contatore a 16 bit implementato nel DIALOG: è possibile di effettuare uno start e uno stop del conteggio utilizzando due impulsi successivi. A seguito di uno start viene incrementato di uno il valore, contenuto nel registro a 16bit del contatore (Rate Counter), ogni volta che il canale fisico, impostato nel registro 26, è attraversato da un segnale impulsivo.

Questa funzionalità è stata testata utilizzando un segnale impulsivo periodico prodotto da un generatore d’impulsi e effettuando un “Software_start_stop” dal menù dell’Actel. Si è verificata così l’esattezza del conteggio.

Per praticità d’utilizzo, la lettura e l’azzeramento dei registri del contatore, sono accessibili direttamente dal menu principale relativo al DIALOG (Dialog menu, fig. 5.41).

Nella modalità “Pattern generator” il segnale test_pulse è usato come condizione per l’invio dei bit di pattern. Se facciamo corrispondere ogni bit dei registri di pattern ad un canale fisico, ogni volta che si riceve un impulso tutto avviene come se si ricevesse un hit su ogni canale fisico con pattern uno. Tale caratteristica ha lo scopo di permettere un facile test dell’apparato.

Anche questa funzionalità è stata testata con pattern diversi studiando i canali logici in uscita con un Logic Analizer.

Se si vuole calibrare il DLL si accede ad un’ulteriore schermata in cui si possono scegliere diverse operazioni.

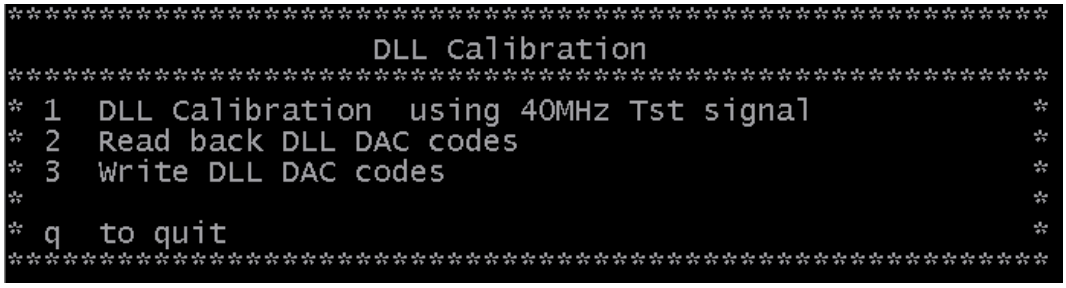


Figura 5.43: Menu per la calibrazione del DLL del DIALOG.

La funzione principale è la prima, "DLL Calibration using 40 MHz Tst signal", che permette di effettuare la calibrazione utilizzando, come riferimento, un segnale periodico di 40 MHz inviato lungo test_pulse. Questo è necessario perché il DIALOG deve assolvere la funzione di allineare temporalmente i segnali provenienti dai canali fisici che ne costituiscono l'ingresso. Infatti, i canali fisici, pur trasportando un segnale generato in maniera sincrona con i BX, subiscono percorsi diversi all'interno del rivelatore e necessitano quindi di un riallineamento. A questo fine ogni ingresso dispone di un ritardo controllato tramite la struttura DLL-ADC mostrata in figura 5.44.

In breve, gli otto canali fisici provenienti da un ASD, vengono riallineati utilizzando una tensione di riferimento (delay control voltage level) prodotta da un convertitore digitale-analogico. Quando si opera la calibrazione il blocco DLL implementato nel DIALOG incrementa, singolarmente e un passo alla volta, il ritardo degli 8 canali fisici di uno dei suoi due blocchi d'ingressi, uno per ogni ASD, utilizzando come riferimento il clock a 40 MHz inviato sulla linea test_pulse dall'Actel. Quando sono allineati temporalmente un convertitore analogico-digitale (ADC) a 8bit campiona il "delay control voltage level" e ne scrive il valore in uno dei due registri adibiti (21 e 22). Questo codice di riferimento verrà usato dal convertitore digitale-analogico per generare il "delay control voltage level" usato da tutti ritardi.

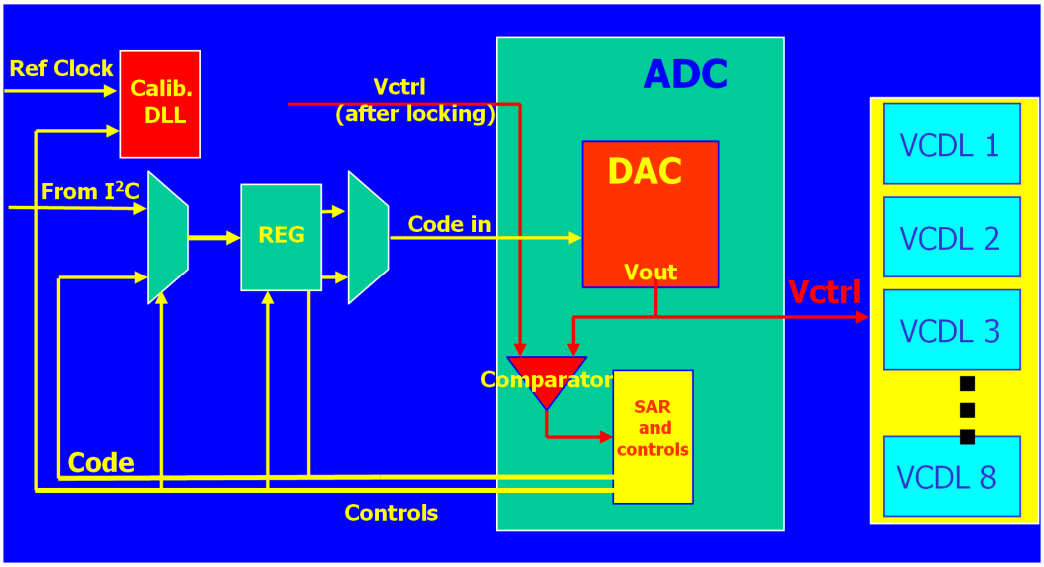


Figura 5.44: Struttura DLL-ADC per il riallineamento temporale dei canali fisici.

La giusta procedura di calibrazione è la seguente:

1. si imposta a 3 il registro Test_mode ("DLL calibration").
2. si impostano i multiplexer dell'Actel in modo che sia inviato il clok40 sulla linea test_pulse.
3. si disabilita la linea test_pulse nel DIALOG scrivendo 0 in Test_mode
4. si disabilita l'invio del clok40 da parte dell'Actel.

Questa procedura, abbastanza macchinosa da eseguire a mano, è stata automatizzata nella funzione "DLL Calibration".
Le altre due funzionalità accessibili dal menu permettono rispettivamente la lettura e la scrittura manuale dei codici di riferimento.

Main DLL characteristics	
Locking time	< 500 ns
Period range for correct locking	18 ns ÷ 40 ns (55 MHz ÷ 25 MHz)
Nominal reference period	25 ns (40 MHz)
DNL @ 40 MHz	± 0.3 LSB

Tabella 5.5: Caratteristiche principali del DLL del DIALOG.

Main ADC characteristics	
Resolution	8 bit
Architecture	SAR
Conversion time	< 2 μs
Size	442 x 178 μm^2

Tabella 5.5: Caratteristiche del convertitore analogico-digitale del sistema di calibrazione.

Poiché la calibrazione del DLL va eseguita ad ogni accensione del sistema, nel menu principale del DIALOG (fig. 5.41) è stata messa una scorciatoia per accedere alle funzioni di calibrazione.

Sempre dal menu principale, è possibile effettuare il test automatico dei due convertitori digitale-analogico (DAC) (uno per ogni ASD), implementati nel DIALOG, allo scopo di fornire delle tensioni di soglia per le schede ASD. Il test consiste nel fare in modo che il convertitore selezionato produca un "dente di sega" incrementando la tensione prodotta, da 0 fino al suo limite superiore (2,5Volt), un passo alla volta e ripetendo questo procedimento dall'inizio ogni volta che si seleziona il test. Questo è realizzato incrementando progressivamente il valore contenuto nel registro del DAC in questione (19 o 20).

Nella tabella 5.6 sono riportate le caratteristiche dei due convertitori digitale-analogico. Il test è stato effettuato registrando con l'oscilloscopio l'incremento della tensione di soglia, riportato in figura 5.45.

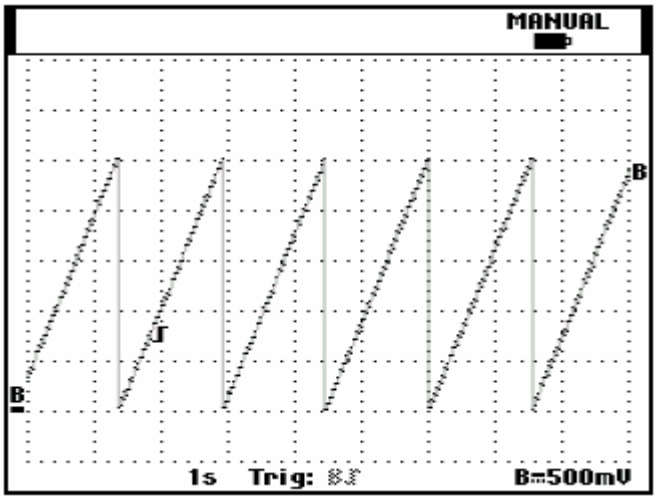


Figura 5.45: Dente di sega prodotto dal test dei due DAC per le soglie per le schede ASD.

Main DAC characteristics	
Resolution	8 bit
Area	146 x 153 μm^2
Supply voltage	2.5 V
DNL	± 0.25 LSB
INL	± 0.25 LSB
Power consumption	500 μW (average)
Output resistance	≈ 20 kOhm (code dependent)
Settling time (@0pF load)	<50 ns
Settling time (@5pF load)	< 250 ns
V output	(VRP – VRN)/256 x code

Tabella 5.6: Caratteristiche dei due DAC implementati nel chip DIALOG.

6. La stazione di test delle camere MPWC

6.1 La stazione di test e la misura dell'efficienza e del jitter delle camere a fili dei rivelatori di muoni.

Ai fini del trigger delle particelle in LHCb è necessario che ogni particella venga rivelata in una finestra temporale inferiore al periodo di bunch crossing (25 ns).
Per soddisfare questo requisito si richiede che i rivelatori delle stazioni per i muoni (M1-M5) abbiano una buona risoluzione temporale in modo da assegnare univocamente il BX identifier ad ogni particella. In particolare si vuole che il sistema abbia un'efficienza del 95% in una finestra di 20 ns.
Per ottenere la risoluzione necessaria ogni MWPC (Multi Wire Proportional Chamber) è composta da quattro gap messe in OR in modo che il ritardo tra il passaggio della particella ionizzante e la sua rivelazione da parte della camera venga statisticamente ridotto. Infatti, il ritardo di una singola gap, è distribuito secondo una curva di probabilità con larghezza a metà altezza di circa 20 ns, non sufficiente ai nostri scopi visto che comporterebbe una percentuale di particelle nella finestra di misura ben più bassa del 95%. L'Or logico di più camere permette di restringere tale larghezza a circa 3ns, sufficiente ad assicurare l'efficienza richiesta. La larghezza a metà altezza della distribuzione dei ritardi è detta jitter della camera.

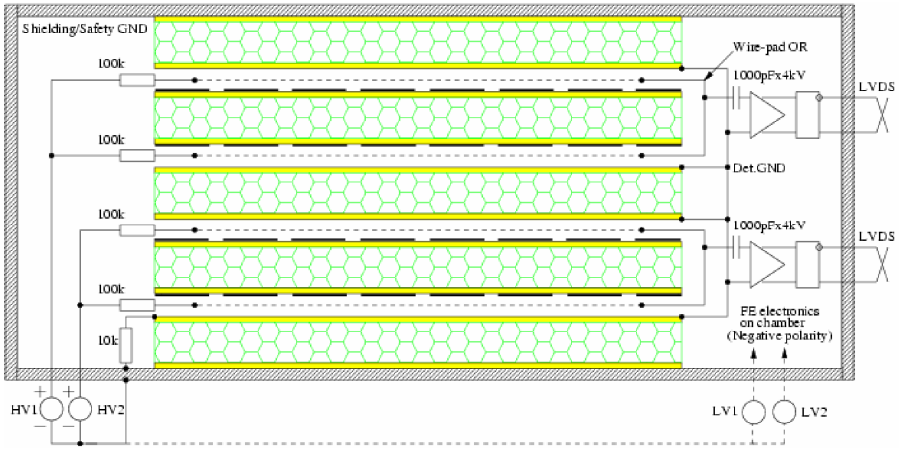


Figura 6.1: Or delle gap delle MWPC

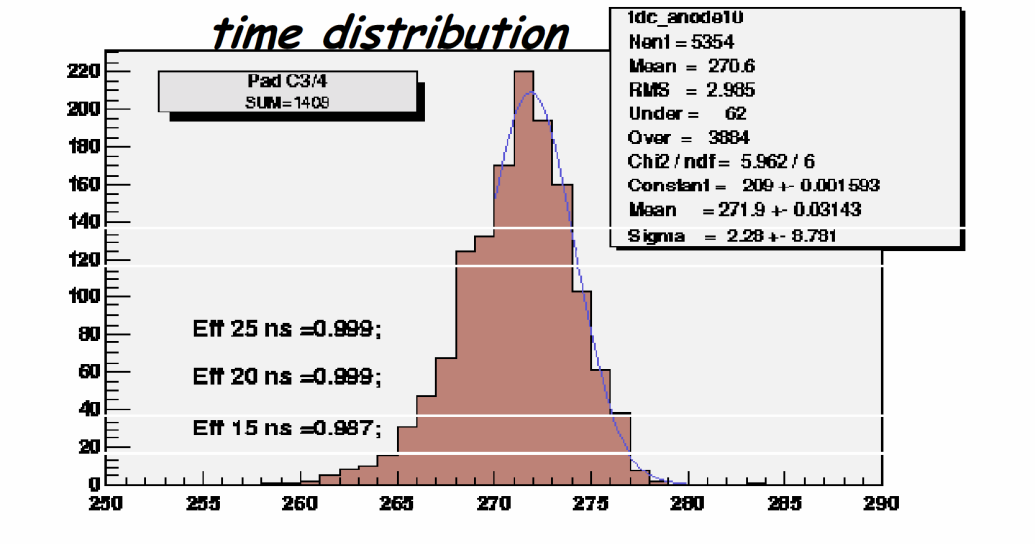


Figura 6.2: distribuzione temporale del ritardo di un MWPC.

Nel test delle camere risulta evidente la necessità di caratterizzarne il jitter in modo da assicurarne la risoluzione temporale e l'efficienza: a questo scopo verrà realizzata una stazione di test che utilizzerà, come fonte di particelle ionizzanti, i raggi cosmici. Il progetto è quello di utilizzare l'apparato mostrato in figura 6.3 [Ref.30]. Sulla stazione di test vengono montate sei MWPC contemporaneamente, sia per ridurre i tempi di test, sia per ottimizzare il tracking muonico, necessario

per il funzionamento della stazione. Le MWPC sono disposte tra due scintillatori che effettuano il trigger delle particelle ionizzanti e ne riveleranno il momento d'entrata e d'uscita dall'apparato. Il sistema così costituito è esposto ai raggi cosmici e i canali di uscita sono collegati ad un TDC (Time to Digital Converter) che registra il momento di passaggio delle particelle nelle camere o, più propriamente, il momento in cui la traccia della particella viene rivelata dalle camere. La differenza fra la misura effettuata dagli scintillatori e quella delle camere permette di calcolare il ritardo della rivelazione e di dare una stima del jitter. Un ulteriore scintillatore, posto in fondo alla stazione, sotto uno scudo di piombo, messo in AND logico con i primi due, permette di filtrare il fondo dei raggi cosmici e selezionare solo le particelle con energia nell'ordine del GeV. La qualità della misura del jitter dipende fortemente dall'accuratezza con cui si riesce a determinare l'istante iniziale di passaggio T_0. Il ritardo medio di ogni scintillatore è stimabile intorno a 1ns ma a questo va aggiunto il tempo di propagazione del segnale dal punto di passaggio della particella al fotomoltiplicatore che ne registra la presenza. Considerando la lunghezza dello scintillatore di circa 1,5 m e che, in esso, la velocità della luce vale circa $2x10^8$m/s, questo ritardo può arrivare fino a 7 ns : per rendere significativa la misura del jitter deve essere opportunamente corretto. La correzione avviene utilizzando il software di tracking: al passaggio della particella nella prima camera si apre una finestra di ricerca di 20 ns intorno alla pad accesa e si passa a considerare la MWPC sottostante. Così facendo per tutte e sei le camere è possibile risalire, con precisione sufficiente, all'angolo di incidenza della particella e ottenere il tempo T_0 di entrata e T_f di uscita a meno di 1ns.

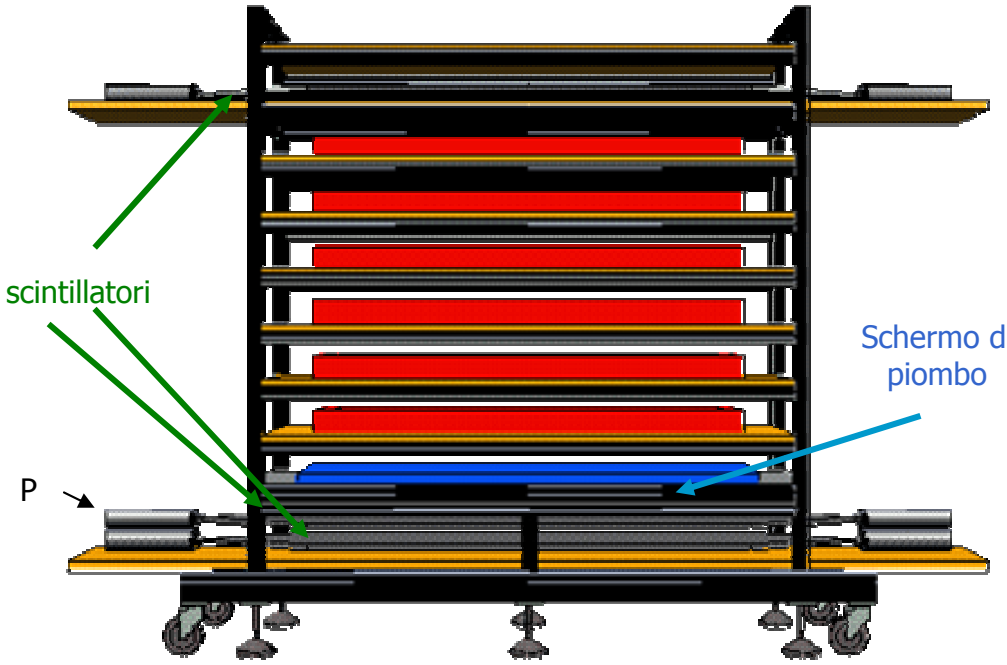


Figura 6.3 : La stazione di test: Il trigger è effettuato da 3 scintillatori con doppia lettura: è stata misurata un'efficienza totale del 90% e una risoluzione temporale di ~1.8 ns.
Lo strato di piombo permette invece di eliminare gli sciami di particelle a bassa energia.

6.2 Lo scopo del multiplexer

Per l'acquisizione dei dati provenienti dalla stazione di test è necessario disporre di un TDC (Time to Digital Converter) capace di monitorare tutti i canali fisici provenienti dalle sei MWPC che vengono testate contemporaneamente. Ciascuna camera dispone di 96 canali di uscita e quindi per poter servire tutta la stazione di test sarebbe necessario un TDC con 6x96=576 ingressi oppure l'utilizzo di più strumenti in parallelo. Inoltre, perché la misura del jitter sia valida, è necessaria una risoluzione minima di 1ns. Un TDC con queste caratteristiche ha un costo elevato e l'acquisto di un apparato capace di servire tutti i canali inciderebbe in

maniera significativa sul costo del test. Si è quindi pensato di utilizzare un solo "TDC Caen V767" a 128canali riducendo i 576canali della stazione di test ai 128 disponibili tramite una scheda multiplexer e ritardando diversi canali che interessano lo stesso ingresso di tempi via via crescenti così da renderli distinguibili fra loro. La realizzabilità di questa soluzione scaturisce essenzialmente da due caratteristiche del sistema: il basso rate di particelle ionizzanti nei raggi cosmici e il lungo range temporale di funzionamento di cui è dotato il TDC Caen V767.

Il basso rate di particelle ionizzanti a energia elevate, che può essere stimato intorno a 1particella/minuto·cm^2, rende improbabile la sovrapposizione di hit di canali diversi corrispondenti al passaggio di particelle diverse, senza imporre limiti restrittivi all'acquisizione di un singolo evento.

Il range di funzionamento del TDC, invece, rende possibile ritardare i canali di un tempo tale da assicurare la non sovrapposizione di canali diversi al passaggio di una particella: perché ciò non avvenga il ritardo Δt tra un canale e il seguente dovrà essere molto maggiore della risoluzione temporale richiesta alle MWPC. Si è scelto di fissare Δt intorno ai 100ns in modo da assicurare questa condizione.

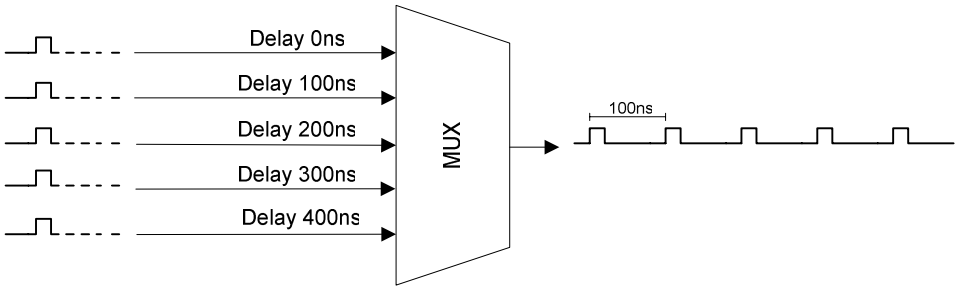


Figura 6.4: funzione del multiplexer con delay

Esiste un ulteriore problema dal punto di vista economico: l'acquisto di tante linee di ritardo quanti sono i canali uscenti dalla stazione di test con una precisione tale da non influire sulla risoluzione del sistema comporterebbe una spesa tanto elevata da rendere comunque dispendioso il sistema di acquisizione.

Si è così pensato di realizzare un multiplexer, con ritardo sui singoli canali, su di un chip FPGA (Field Programmable Gate Array) di grandi dimensioni, contenente cioè un gran numero di celle elementari, ottenendo i ritardi con l'impiego di funzioni logiche che diano come risultato l'identità:così il segnale "perde tempo" nell'elettronica del chip e si presenta in uscita con il ritardo accumulato. Questa soluzione è risultata essere funzionale ed economica, visto il costo ridotto di FPGA, anche con un gran numero di celle, di nuova generazione. Inoltre ha permesso di inserire facilmente il multiplexer all'interno dell'apparato sperimentale: la versatilità delle FPGA ha reso possibile sia di adattare gli ingressi e le uscite del multiplexer allo standard voluto (LVDS per gli ingressi e TTL per le uscite) senza aggiungere dei "trasmitter/receivers", sia di poter implementare, sullo stesso chip, un "bus i2c-slave" per il controllo esterno.

Technical specification table TDC Caen V767	
Dynamic range	20 bit
LSB	0.8 ns
Double hit resolution	10 ns
Input channels, Start	Differential ECL level, 110 Ohm impedance min. FWHM 10 ns
Trigger, Reset, Busy, Clock, Drdy	Differential ECL level, 110 Ohm impedance Min FWHM 25 ns
Integral non linearity	0.3 ns
Differential non linearity	absolute timing: < 15 % relative timing: < 1.5 %
Local buffer	FIFO 32K (TDC to FIFO tranf. rate = 20 MHz)
Clock source	Internal (40 or 60 MHz) / External
Trigger window	From 1 clock cycle to full dynamic range

Tabella 6.1: Caratteristiche del TDC Caen v767.

Le FPGA così implementate andranno a costituire delle schede multiplexer secondo gli standard VME 9U Che potranno essere inserite, insieme alla scheda TDC Caen V767, in un crate VME 9U .

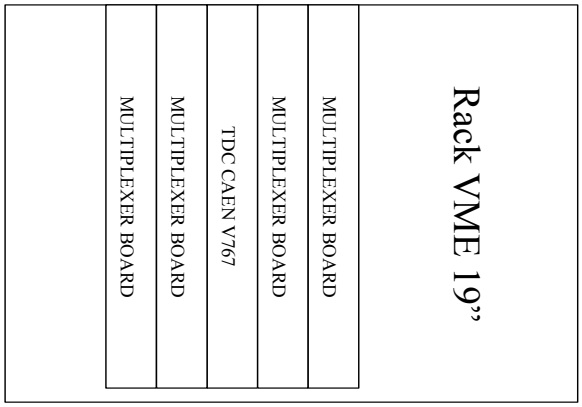


Figura 6.5: Apparato di acquisizione dati

7. Realizzazione del multiplexer

7.1 Scelta della tecnologia

La scelta della tecnologia usata per realizzare il multiplexer, è caduta su FPGA di grandi dimensioni. Un FPGA (Field Programmable Gate Array) è un insieme di celle logiche che comunicano, tra loro e con i loro canali d'ingresso e d'uscita, tramite collegamenti programmabili longitudinali e trasversali. In generale le celle logiche di un FPGA sono molto semplici e funzioni logiche complesse si possono ottenere combinando più celle logiche elementari. Ogni cella comprende un certo numero di "gate", ovvero di canali d'ingresso e d'uscita, fra i quali è possibile stabilire delle connessioni.

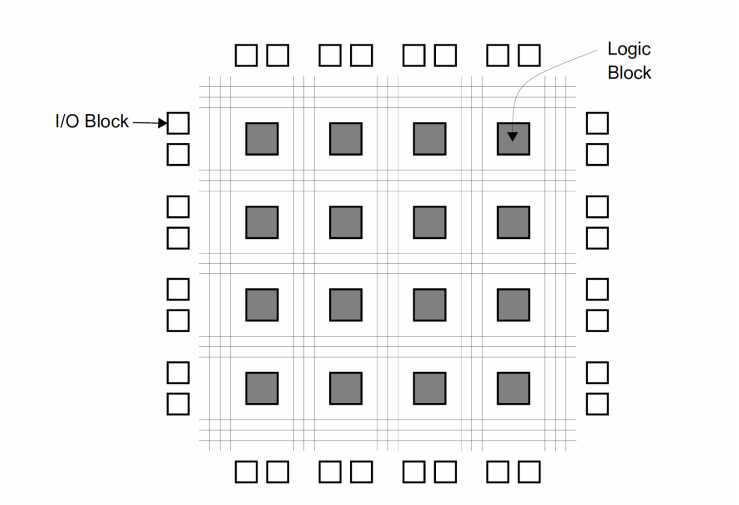


Figura 7.1: schema di una semplice FPGA.

Esistono essenzialmente due tipi di FPGA: **SRAM** (Static RAM) e **antifuse**. Senza soffermarci troppo sul loro funzionamento la differenza principale è che mentre una FPGA antifuse una volta programmata mantiene le connessioni tra le celle definitivamente e non può essere riscritta, una FPGA SRAM può essere

cancellata e riscritta più volte, ma va riscritta ogni volta che gli viene tolta l'alimentazione.[Ref.31]

Si è scelto di utilizzare FPGA di quest'ultimo tipo, perché più versatili, disponibili con un numero maggiore di celle e di gate e riutilizzabili alla fine del test.

Più precisamente il progetto finale è stato realizzato su un FPGA **Xilinx SpartanII XC2S300E** [Ref 32].

La famiglia di FPGA Spartan-IIE risulta la più adatta alla realizzazione del multiplexer perché quella caratterizzata dal migliore rapporto fra prezzo e numero di celle elementari, di gate e di I/O. Supporta ben 19 standard di I/O e si adatta facilmente sia agli standard previsti per la stazione di test che ad eventuali cambiamenti al progetto. Infine la sua architettura, basata su tecnologia Virtex-E, ha permesso di sostituire senza problemi l'FPGA Virtex XCV200, utilizzata nella prima fase della progettazione, con una della famiglia Spartan-IIE e di ritenere indicativi i risultati ottenuti con la prima.

Nella famiglia Spartan-IIE®, la Spartan-II XC2S300E è senza dubbio l'FPGA dove è possibile implementare il maggior numero di ritardi poiché ha la più alta densità di logica: è costituita da 6912 celle logiche e 300.000 gate.

Un altro punto di forza è la possibilità di far funzionare l'elettronica implementata a frequenze superiori a 200 MHz e quindi molto al di sopra delle frequenze usate dall'elettronica del rivelatore LHCb(in particolare la Service Board) che probabilmente sarà usata per il monitor e il controllo del multiplexer durante i test.

È infine possibile effettuare progettazione, mappatura[5], piazzamento [6]e sbroglio[7] dell'FPGA tramite il software "Xilinx ISE development system 4.2", in nostro possesso, che permette l'automatizzazione di molte operazioni.

[5] È l'operazione con cui il software trascrive la logica nelle celle.
[6] È l'operazione con cui si distribuisce nella maniera ottimale la logica nell'FPGA.
[7] È il processo con cui si determinano i collegamenti fra i vari gate delle celle.

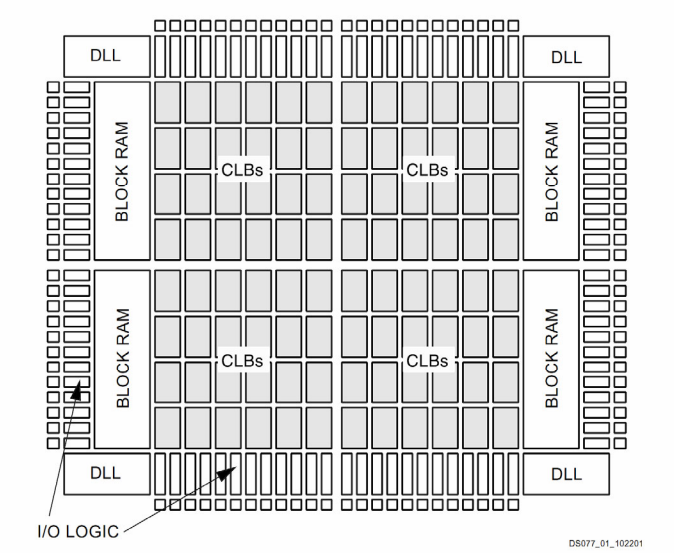


Figura 7.2: diagramma a blocchi di un FPGA della famiglia SPARTA-IIE.

I test effettuati hanno messo in evidenza la realizzabilità di questo progetto, sia per quanto riguarda la stabilità dei ritardi, sia per il jitter introdotto dalla scheda multiplexer che, per quanto detto nel capitolo precedente, gioca un ruolo fondamentale.

Test effettuati sulla Xilinx Virtex XCV200 hanno prodotto valori del jitter al di sotto dei 400 ps. Le misure sono state eseguite inviando un segnale periodico a 10 MHz come ingresso per una versione semplificata del multiplexer, implementata sulla scheda Virtex, e studiando la funzione di distribuzione dei ritardi attraverso un oscilloscopio "TLC CSA 803A" con funzioni di analisi statistica. L'oscilloscopio ha fornito i valori "picco-picco" della distribuzioni dei ritardi, ovvero la distanza tra il valore più grande e quello più piccolo misurato.

In tabella 7.1 sono riportati i risultati delle misure effettuate variando la tensione di alimentazione, entro i limiti di funzionamento, a temperatura ambiente e scaldando l'FPGA con una pistola termica. I piccoli valori ottenuti non hanno richiesto misure più accurate.

Conteggi	RMSD(ps)	PkPk(ps)	Vccint(V)	Temperatura
100000	151	798	2,5	Ambiente
100000	169,9	886	2,625	Ambiente
100000	151,5	884	2,375	Ambiente
100000	124,27	824	2,5	~ 50°C
100000	112,27	730	2,625	~ 50°C
100000	102,8	692	2,375	~ 50°C

Tabella 7.1: Risultati delle misure di jitter effettuate sulla Virtex XCV200. La distanza picco-picco (PkPk) costituisce un limite maggiore per il valore del jitter.

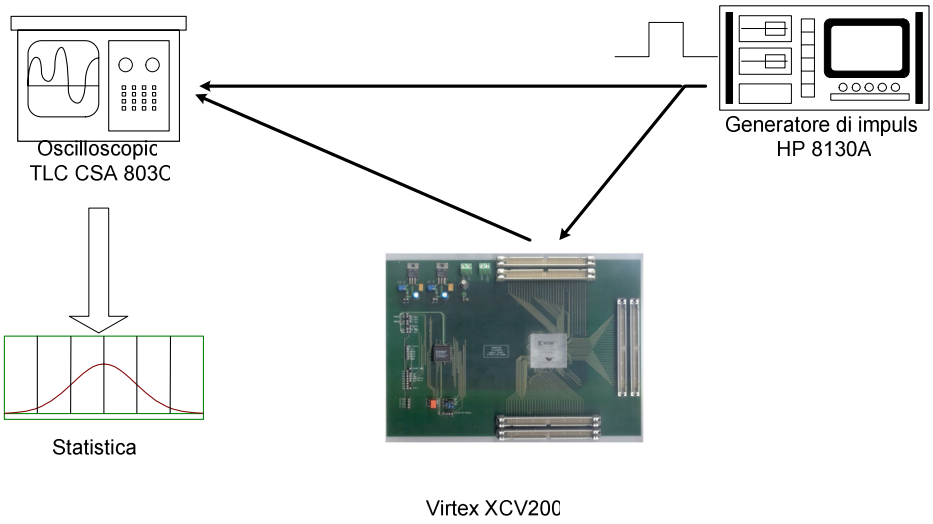


Figura 7.4: Apparato di misura del jitter dell'FPGA Virtex XCV200.

Quanto alla possibilità di implementare ritardi nell'FPGA si noti che, anche se si cerca di ripetere la stessa struttura, cioè lo stesso sistema di celle, più volte il piazzamento e lo sbroglio delle celle avvengono in modo automatico e con percorsi diversi fra un gate ed il successivo: perciò si ottengono ritardi variabili, anche se in un range molto ristretto. In particolare si è deciso di implementare i ritardi su ogni linea utilizzando una o più unità di ritardo da 100ns, sufficienti ad assicurare la non sovrapposizione dei segnali di canali diversi: per ogni blocco di ritardo si è misurato un valore compreso tra 90ns e 105ns.

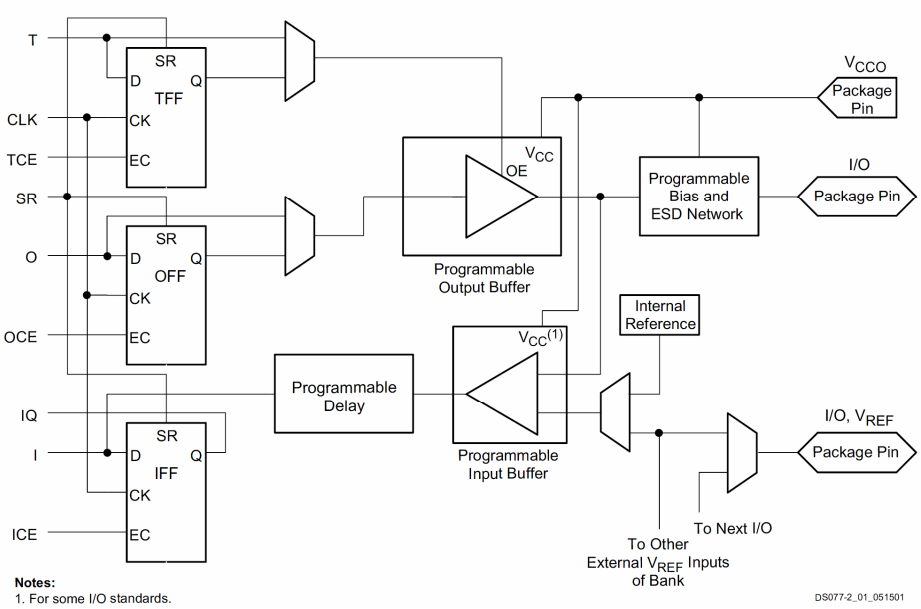


Figura 7.3: diagramma di un blocco di I/O della Sparten-IIE.

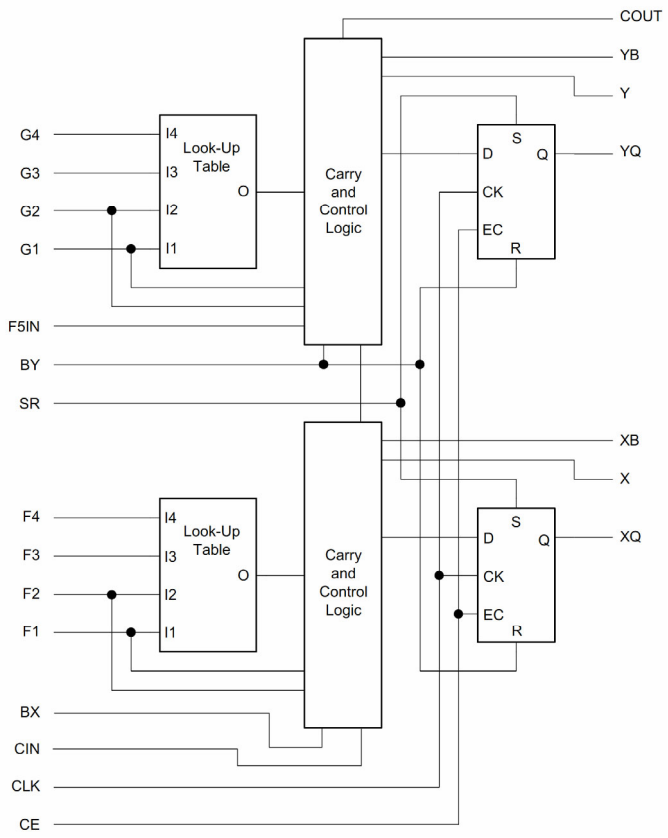


Figura 7.5: diagramma a blocchi di una cella elementare della Sparten-IIE. Ogni cella contiene due blocchi di logica identici che, per ottenere il maggior ritardo possibile, sono stati utilizzati singolarmente entrando e uscendo due volta dalla stessa cella.

La variabilità del ritardo non costituisce comunque un problema poiché basterà calibrare l'apparato un'unica volta per essere sicuri di conoscere con esattezza il ritardo di ogni canale.

7.2 Funzionamento

Il multiplexer finale è stato progettato con ingressi ritardati di 0, 100, 200, 300 e 400 ns, in modo da poter ridurre, con ogni chip, 36 canali di ingresso in 8 di uscita. Il multiplexer 36x8 è realizzato implementando sull'FPGA otto multiplexer ad un solo canale di uscita così distribuiti:

- 4 con quattro canali di ingresso, ognuno ritardato di ~100ns (nella precisione data prima) rispetto al precedente, e uno di uscita. Il ritardo massimo sarà quindi di 300ns.
- 4 con cinque ingressi, ritardati secondo lo schema precedente, e un uscita. Questa volta il ritardo massimo sarà di 400ns.

Il multiplexer dispone quindi di 8 ingressi con ritardi da 0, 100, 200 e 300 ns e solo 4 con un ritardo pari a 400 ns. Conseguentemente le uscite non sono tutte uguali: le prime quattro coprono quattro ingressi e, le restanti quattro, cinque ingressi.
Si è cercato di mantenere questa modularità 36x8 in previsione di inserire, in una scheda formato VME 9U, quattro Spartan-IIE così programmate ottenendo schede multiplexer 144x32 come richiesto dalla modularità del sistema.
Come gia detto le sei camere della stazione di test disporranno un massimo, dipendente dalla stazione e dalla zona a per cui sono progettate, di 96x6=576 canali fisici. Perché sia possibile servire tutti i canali con quattro schede multiplexer sarà necessario che ogni scheda abbia proprio 576/4=144 ingressi. Allo stesso modo ogni scheda dovrà avere 128/4=32 uscite e questo è linea con la scelta effettuata.

Inoltre, la modularità 36x8 per ogni FPGA, è in ottimo accordo con il numero massimo di blocchi di ritardo da 100 ns implementabili nella Spartan-II XCV300E : questa scelta massimizza l'utilizzo delle risorse dell'FPGA e permette così di non superare il numero massimo di chip che si possono sistemare su una scheda VME. Una modularità diversa sarebbe risultata poco pratica: l'utilizzo di più schede VME avrebbe aumentato i problemi di cablaggio mentre l'impiego di FPGA con un più alto numero di celle logiche avrebbe comportato un notevole aumento dei prezzi, essendo la XCV300E la più grande della famiglia Spartan-IIE.

Il sistema di acquisizione dati, schede multiplexer più TDC, e le camere dovranno essere cablate secondo lo schema riportato in fig 7.6. [Ref.33]

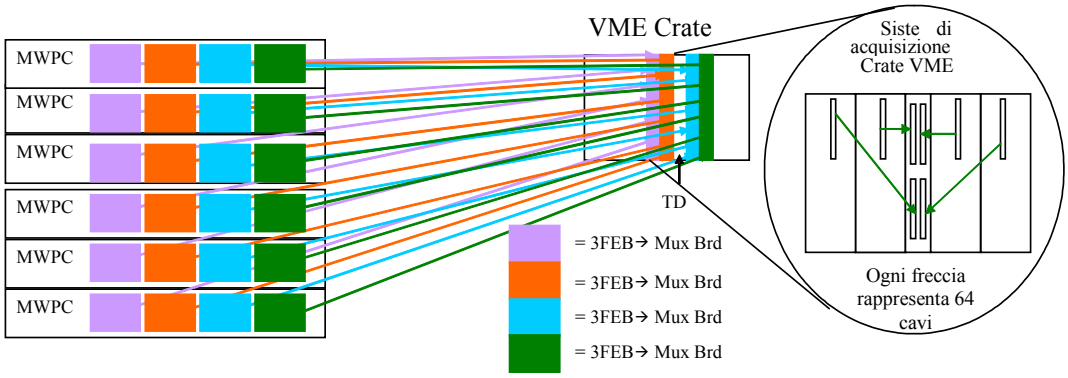


Figura 7.6: Cablaggio della stazione di test.Ogni freccia colorate rappresenta 64 cavi. Ogni MWPC presenta un massimo di 96 canali fisici. I colori diversi delle frecce indicano a quelle scheda multiplexer è indirizzata.

Su ogni FPGA sono stati inseriti anche 36 bit di maschera, messi in AND con ogni canale di ingresso, che permettono di abilitare o disabilitare i singoli canali (1=enable;0=disable) in modo da poterli osservare singolarmente per individuare eventuali malfunzionamenti. Per controllare i bit di maschera, è stato inserito un modulo i2c slave che permette la comunicazione, su i2c bus, fra il multiplexer e l'elettronica di controllo, probabilmente costituita dalla Service Board. Per lo

stesso scopo sono stati riservati altri quattro bit per contenere l'indirizzo i2c dell'FPGA mentre i tre mancanti saranno forniti da registri esterni in modo da costituire il prefisso della scheda.

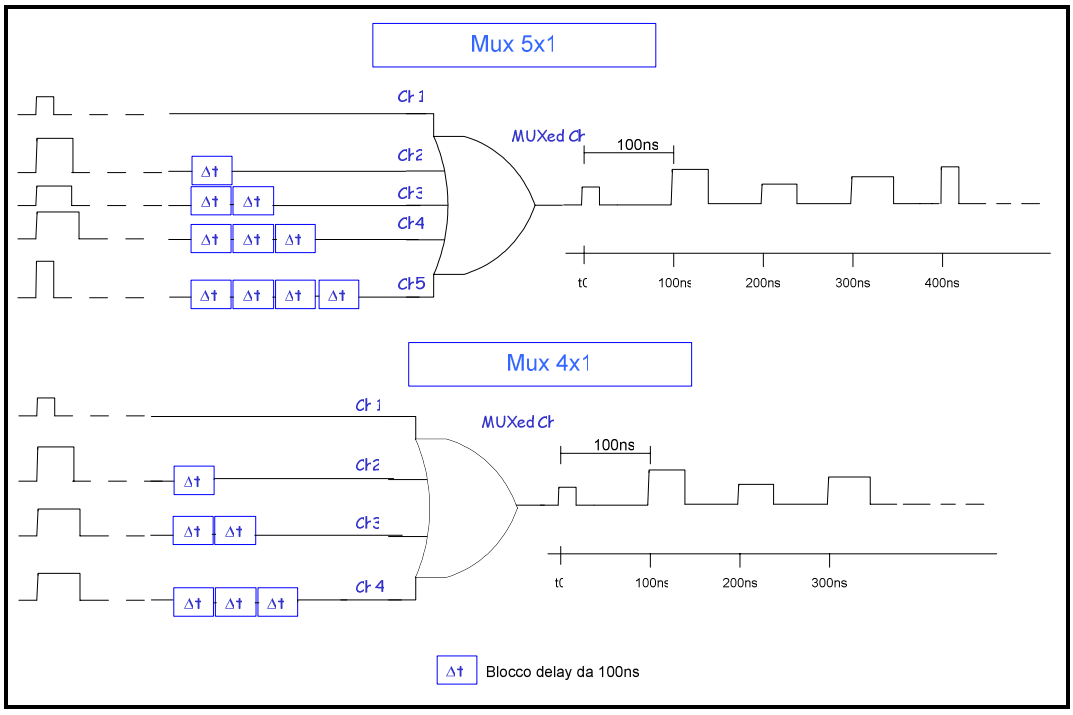


Figura 7.7: funzionamento dei multiplexer 5x1 e 4x1 che costituiscono il Multiplexer36x6.

I bit di maschera sono correlati ai 36 ingressi del multiplexer36x8 in maniera tale che, se si considerano i 5 registri a 8 bits come un unico registro a 40 bits, il numero del bit di registro corrisponde al nome del canale di ingresso a cui si riferisce secondo lo schema indicato in tabella 7.2.

Tutti gli ingressi dei multiplexer sono coppie differenziali nello standard LVDS, come richiesto dalla struttura delle MWPC, mentre i canali di uscita seguono lo standard TTL, come richiesto dal TDC Caen V767.

I ritardi aspettati per ogni ingresso sono stati calcolati usando l'applicazione "Timing Analyzer" dello "Xilinx ISE development system 4.2" che ha fornito i

valori teorici per il ritardo massimo di ogni canale, riportati in tabella 7.3 insieme a quelli nominali. Le variazioni da tale valore sono contenute entro il 10%.

canale di ingresso	sottoindirizzo i2c	bit	Canale di ingresso	indirizzo i2c	bit
In0	0	xxxxxxx1	In18	2	xxxxx1xx
In1	0	xxxxxx1x	In19	2	xxxx1xxx
In2	0	xxxxx1xx	In20	2	xxx1xxxx
In3	0	xxxx1xxx	In21	2	xx1xxxxx
In4	0	xxx1xxxx	In22	2	x1xxxxxx
In5	0	xx1xxxxx	In23	2	1xxxxxxx
In6	0	x1xxxxxx	In24	3	xxxxxxx1
In7	0	1xxxxxxx	In25	3	xxxxxx1x
In8	1	xxxxxxx1	In26	3	xxxxx1xx
In9	1	xxxxxx1x	In27	3	xxxx1xxx
In10	1	xxxxx1xx	In28	3	xxx1xxxx
In11	1	xxxx1xxx	In29	3	xx1xxxxx
In12	1	xxx1xxxx	In30	3	x1xxxxxx
In13	1	xx1xxxxx	In31	3	1xxxxxxx
In14	1	x1xxxxxx	In32	4	xxxxxxx1
In15	1	1xxxxxxx	In33	4	xxxxxx1x
In16	2	xxxxxxx1	In34	4	xxxxx1xx
In17	2	xxxxxx1x	In35	4	xxxx1xxx

Tabella 7.2: Indirizzi dei bits di maschera degli ingressi di un multiplexer36x8. La prima colonna riporta il nome del canale di ingresso. La seconda il sottoindirizzo i2c che deve seguire l'indirizzo i2c del multiplexer. L'ultima colonna riporta la posizione del bit che abilita il canale.

Il pin-out della FPGA, riportato in appendice C, presenta, oltre a 36 canali di ingresso e gli 8 di uscita, i canali necessari per la comunicazione su i2c-bus. Il modulo i2c-slave è lo stesso utilizzato per l'ACTEL della Service Board opportunamente adattato al multiplexer, è sincrono e necessita che sulla scheda multiplexer sia inserito un opportuno clock per il suo funzionamento. Per la comunicazione su i2c-bus è stato scelto lo standard TTL.

Ingressi	Uscite	ritardo nominale (ns)	Ritardo calcolato (ns)
in<0>	out_p<0>	0	11,265
in<1>	out_p<0>	100	110,9
in<2>	out_p<0>	200	207,259
in<3>	out_p<0>	300	307,146
in<4>	out_p<0>	400	403,482
in<5>	out_p<1>	0	10,843
in<6>	out_p<1>	100	109,376
in<7>	out_p<1>	200	211,686
in<8>	out_p<1>	300	301,614
in<9>	out_p<1>	400	397,372
in<10>	out_p<2>	0	13,206
in<11>	out_p<2>	100	115,685
in<12>	out_p<2>	200	209,739
in<13>	out_p<2>	300	306,149
in<14>	out_p<2>	400	405,611
in<15>	out_p<3>	0	13,25
in<16>	out_p<3>	100	112,23
in<17>	out_p<3>	200	207,755
in<18>	out_p<3>	300	308,718
in<19>	out_p<3>	400	404,342
in<20>	out_p<4>	0	13,962
in<21>	out_p<4>	100	111,67
in<22>	out_p<4>	200	207,929
in<23>	out_p<4>	300	311,034
in<24>	out_p<5>	0	13,415
in<25>	out_p<5>	100	111,672
in<26>	out_p<5>	200	210,049
in<27>	out_p<5>	300	309,361
in<28>	out_p<6>	0	14,317
in<29>	out_p<6>	100	112,108
in<30>	out_p<6>	200	209,129
in<31>	out_p<6>	300	306,023
in<32>	out_p<7>	0	11,626
in<33>	out_p<7>	100	112,427
in<34>	out_p<7>	200	209,839
in<35>	out_p<7>	300	308,284

Tabella 7.3: Ritardi aspettati di ogni ingresso del multiplexer 36x8. I valori calcolati secondo le specifiche della Xilinx sono da riferirsi al valore massimo del ritardo. Il valore reale potrà variare intorno al 10% del massimo e quindi resterà comunque al disopra del tempo minimo necessario perché non vi siano sovrapposizioni di segnali di canali diversi.

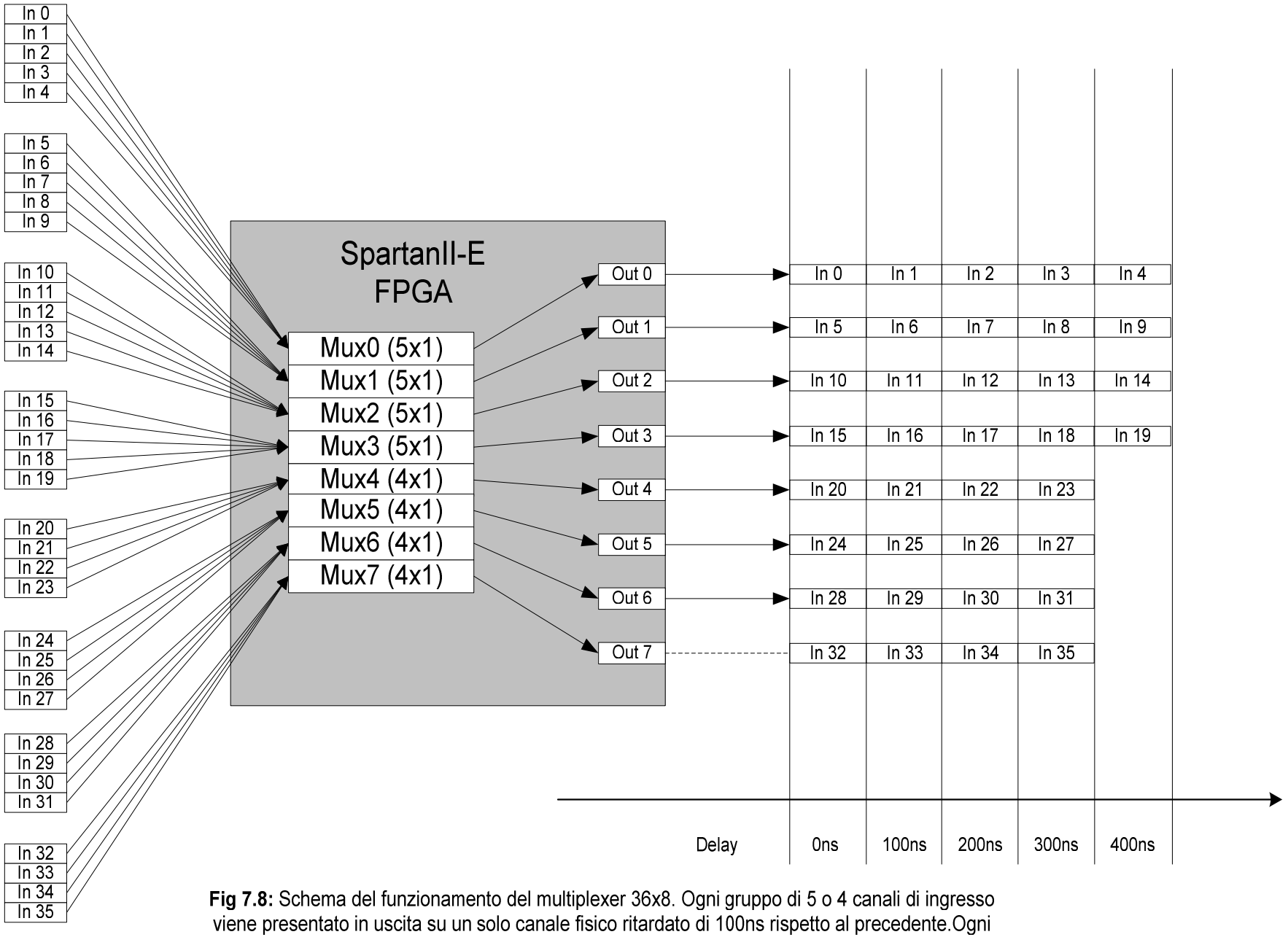


Fig 7.8: Schema del funzionamento del multiplexer 36x8. Ogni gruppo di 5 o 4 canali di ingresso viene presentato in uscita su un solo canale fisico ritardato di 100ns rispetto al precedente.Ogni ingresso va considerato in AND con il relativo registro di maschera.

7.3 Implementazione

Per l'implementazione del multiplexer si è fatto uso delle applicazioni del "Xilinx development system" per la progettazione e lo sbroglio, e del software "ModelSim SE-EE 5.4e" della "Model Technology" per quanto riguarda il test e il debugging.

Il progetto è stato scritto in linguaggio verilog [Ref.34] ed è riportato in appendice.

Il multiplexer 36x8 non è realizzato come un unico progetto ma come la connessione di moduli meno complessi.

La realizzazione è avvenuta seguendo lo schema riportato in figura.

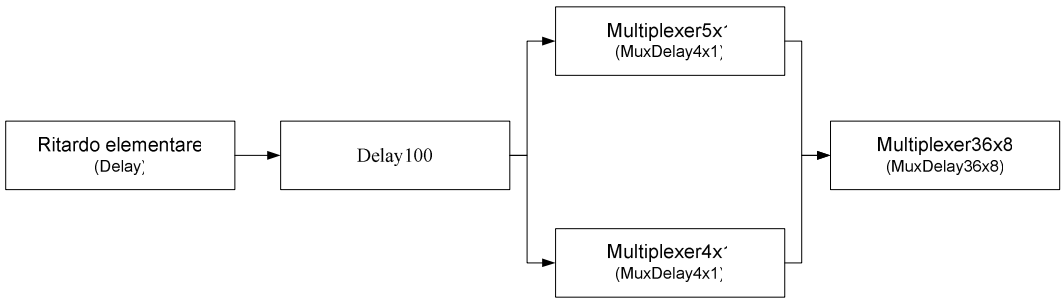


Figura 7.9: struttura del progetto del multiplexer36x8 per la stazione di test.

L'elemento di base del multiplexer è una singola cella di ritardo, cioè una cella dell'FPGA dove il segnale entra e esce immutato,con l'unico scopo di "perdere tempo" nell'elettronica della cella. Questo modulo è stato realizzato interamente a mano, connettendo i singoli gate della cella dell'FPGA e definendone la logica senza l'ausilio dei software di sbroglio a disposizione. La sua progettazione è avvenuta ad un livello più basso di quello del verilog, agendo direttamente sulla logica tramite l'applicazione "FPGA Editor" del software di sviluppo (fig. 7.9)

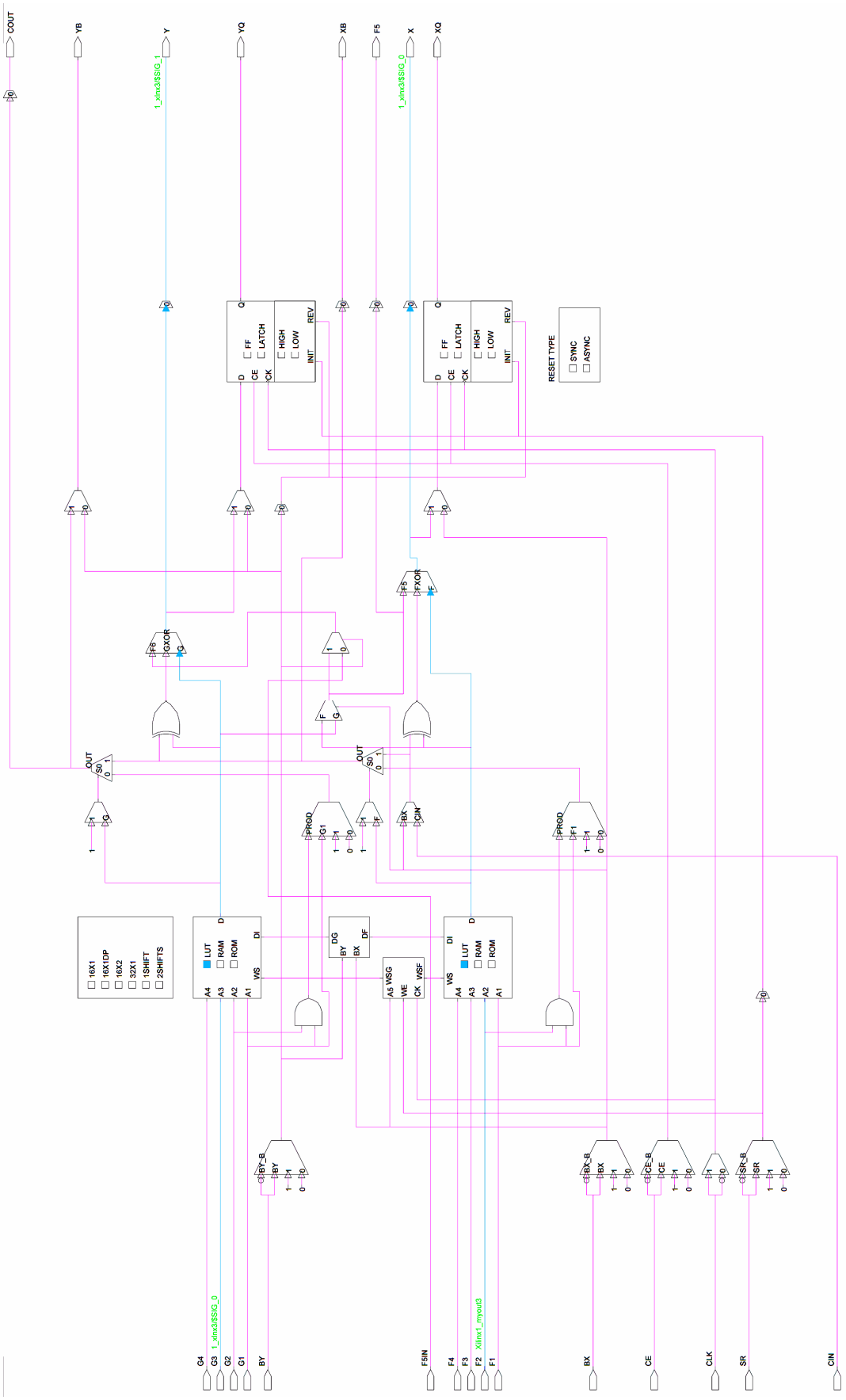


Fig 7.9: Cella elementare di ritardo. Il percorso del segnale è evidenziato in blu. Il segnale entra in G3, arriva al primo blocco di logica dove è implementata la funzione D=A3, esce da Y per rientrare in F2 e, dopo un percorso analogo, uscire da X.

L'implementazione manuale ha permesso di utilizzare ogni cella due volte, sfruttando singolarmente i due blocchi di logica che contiene e collegando l'uscita del primo all'ingresso del secondo (figura 7.9), operazione non possibile utilizzando il software di sbroglio. Otto di queste celle sono state opportunamente collegate per creare una macro con cui è stato realizzato un modulo di ritardo unitario da utilizzare come componente nel resto del progetto.

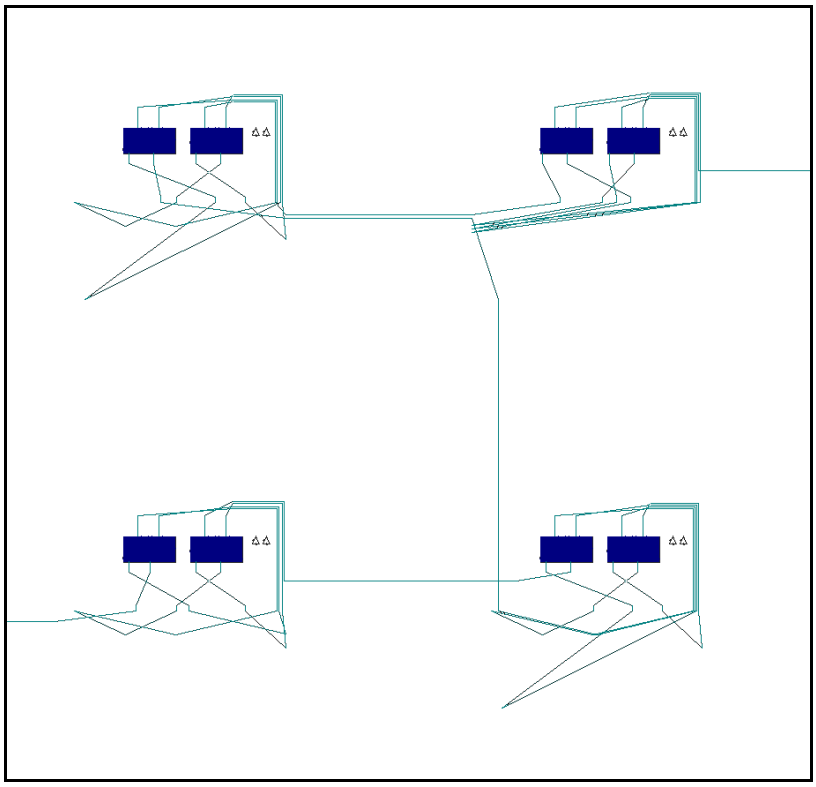


Figura 7.10: Struttura della macro composta da 8 celle utilizzata come modulo di ritardo elementare

Tale componente prende il nome di "delay" e fornisce un ritardo di circa 20 ns. Utilizzando il ritardo elementare si è creato un nuovo componente, questa volta disegnandolo direttamente in verilog, che fornisce un ritardo nominale di 100 ns ed a cui è stato dato il nome di "delay100". Questa operazione ha avuto l'unico

scopo di ottenere un ritardo unitario che rappresenti fisicamente il Δt tra un canale e il successivo.

Utilizzando il delay100 sono stati disegnati i due multiplexer a 5 e a 4 canali di ingresso, MuxDelay5x e MuxDelay4x (fig. 7.7), ognuno con i canali con ritardo incrementato di 100ns fra un ingresso e il successivo, e infine 4 multiplexer per ciascun tipo sono stati posizionati nel progetto principale, MuxDelay36x8, per formare il multiplexer36x8.

Sono stati poi aggiunti i registri di maschera e il driver i2c.

La scelta di ingressi nello standard LVDS ha reso necessario realizzare manualmente il pin-out dell'FPGA, in modo che gli ingressi del multiplexer interessino coppie di pin che supportino tale standard [Ref. 35]. Altra accortezza nella scelta dei pin di I/O è stata quella di scegliere i pin quanto più possibile lontani dal centro dell'FPGA. Infatti i pin della Spartan-II XCV300E sono disposti sotto il chip, come in un processore per PC, secondo il formato "Fine Pitch BGA (FG456)" (figura 7.7), in modo da rendere possibile l'inserimento di un numero maggiore di I/O mantenendo costanti le dimensioni del chip.

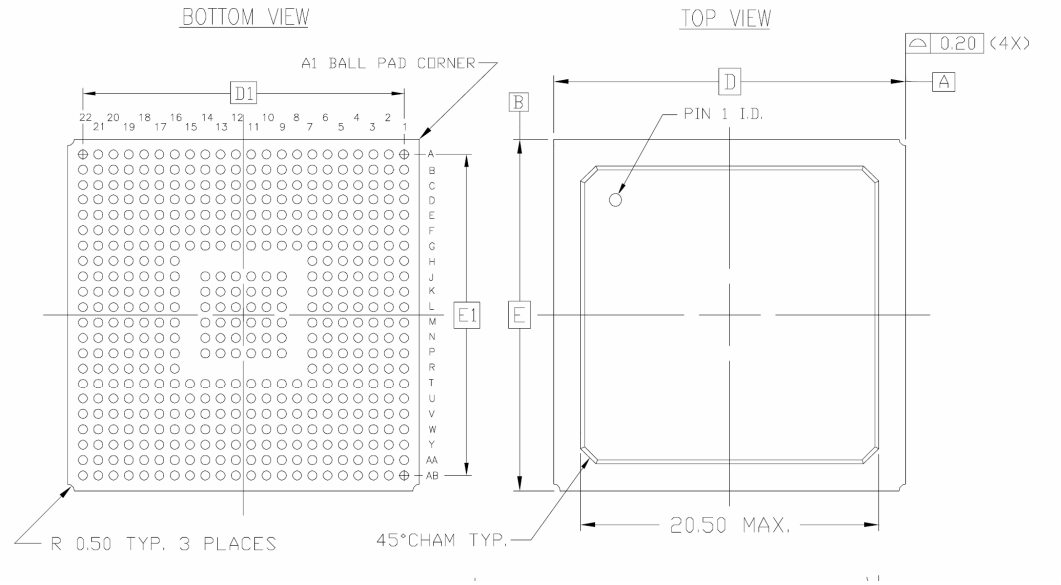


Figura 7.11: Fine Pitch BGA (FG456).

Questo standard introduce una complicanza di carattere tecnico nello sbroglio della scheda su cui verranno piazzate le FPGA: per "far uscire" più pin sulla stessa riga o colonna bisogna aumentare il numero di strati di scheda necessari e, quindi, i costi di realizzazione. Disporre gli I/O quanto più possibile nella zona più esterna risulta evidentemente la scelta migliore.

Il pin-out, riportato in figura 7.12, è stato trascritto nel file "Mux36x8.ucf", con la giusta sintassi, in modo da forzare il software di implementazione a sbrogliare il progetto secondo le direttive volute.

Per completare la fase di progettazione sono stati eseguiti la mappatura, il piazzamento e lo sbroglio dell'elettronica attraverso il "Design Manager" del "Xilinx ISE development system" che ha prodotto i file necessari alla scrittura dell'FPGA. In figura 7.13 è riportata lo schema del multiplexer36x8 implementato sulla Spartan: si vede chiaramente quanto sia stato ottimizzato l'utilizzo delle risorse e la complessità dell'operazione di sbroglio che ha richiesto lunghi tempi di elaborazione da parte del PC. A tale proposito si noti che si è reso necessario realizzare un modulo di ritardo elementare composto da otto celle anziché una, come sarebbe stato conveniente per versatilità e tempi di realizzazione, per semplificare l'operazione di piazzamento e sbroglio automatico, che avrebbe impegnato, altrimenti, più risorse di quante gestibili dalla macchina Windows98 su cui è stato disegnato il multiplexer. Windows 98, infatti, pur avendo un limite superiore alla memoria impaginabile molto alto (~80 Gbyte), non è in grado di gestire più di 2 Gbyte di memoria, fra RAM e memoria virtuale, per ogni applicazione: questa caratteristica costituisce un ostacolo insormontabile per l'implementazione automatica del multiplexer. Anche l'utilizzo di sistemi operativi più prestanti, quali Solaris o WindowsXp, non risolvono il problema a causa della complessità del progetto che risulta comunque troppo elevata. Realizzare una cella elementare più grande ha invece ridotto le variabili del sistema e ha permesso di completare la procedura di ottimizzazione del piazzamento e di sbroglio con un numero minore di iterazioni.

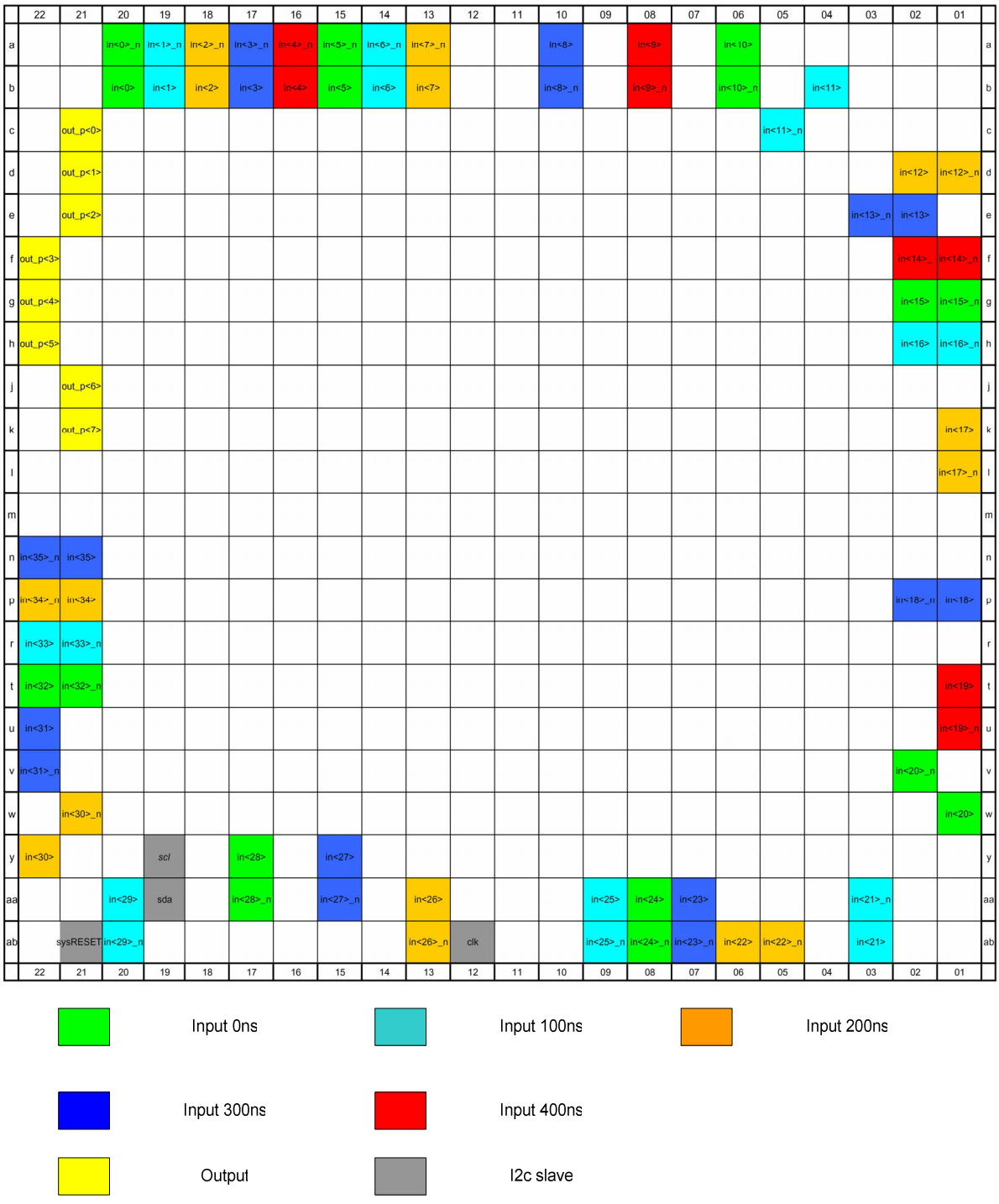


Figura 7.12 : distribuzione dei pin di I/O sul Package Fine Pitch BGA (FG456) della Spartan-II XCV300E.

Sempre utilizzando l' "Xilinx ISE development system 4.2" è stato generato un modello dell'FPGA da utilizzare per la simulazione. Questo modello è costituito da due file: il "MuxDelay36x8_timesim.v", in verilog strutturale, che descrive il multiplexer così come sarà realmente scritto nell'FPGA e contiene informazioni sul ritardo di ogni gate, e il file "MuxDelay36x8_timesim.sdf", che completa le informazioni sull'andamento temporale di un segnale nell'FPGA. Si è così potuti passare ad una fase di simulazione e test del funzionamento del multiplexer, approfondendone la caratterizzazione e verificandone il corretto funzionamento.

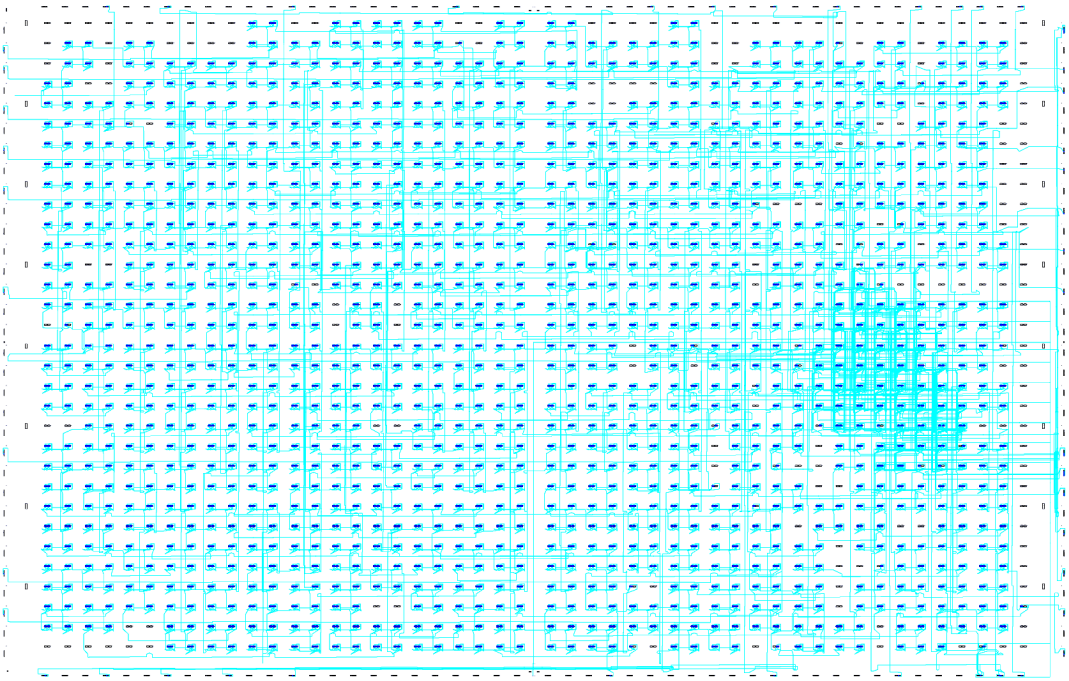


Fig 7.13: Il multiplexer implementato sulla Spartan-II XCV300E. L'addensamento di connessioni, che si può vedere sulla sinistra, è dovuto allo sbroglio del driver i2c.

7.4 Test

Il test del progetto è stato effettuato simulandone il funzionamento tramite il software "ModelSim SE/EE Plus 5.4e" della Model Technology. È stato realizzato in verilog un banco di test dove collegare tra loro il multiplexer 36x8 implementato sulla Spartan-IIE (rappresentato dai file di simulazione) e altri semplici moduli, come mostrato in figura 7.14.

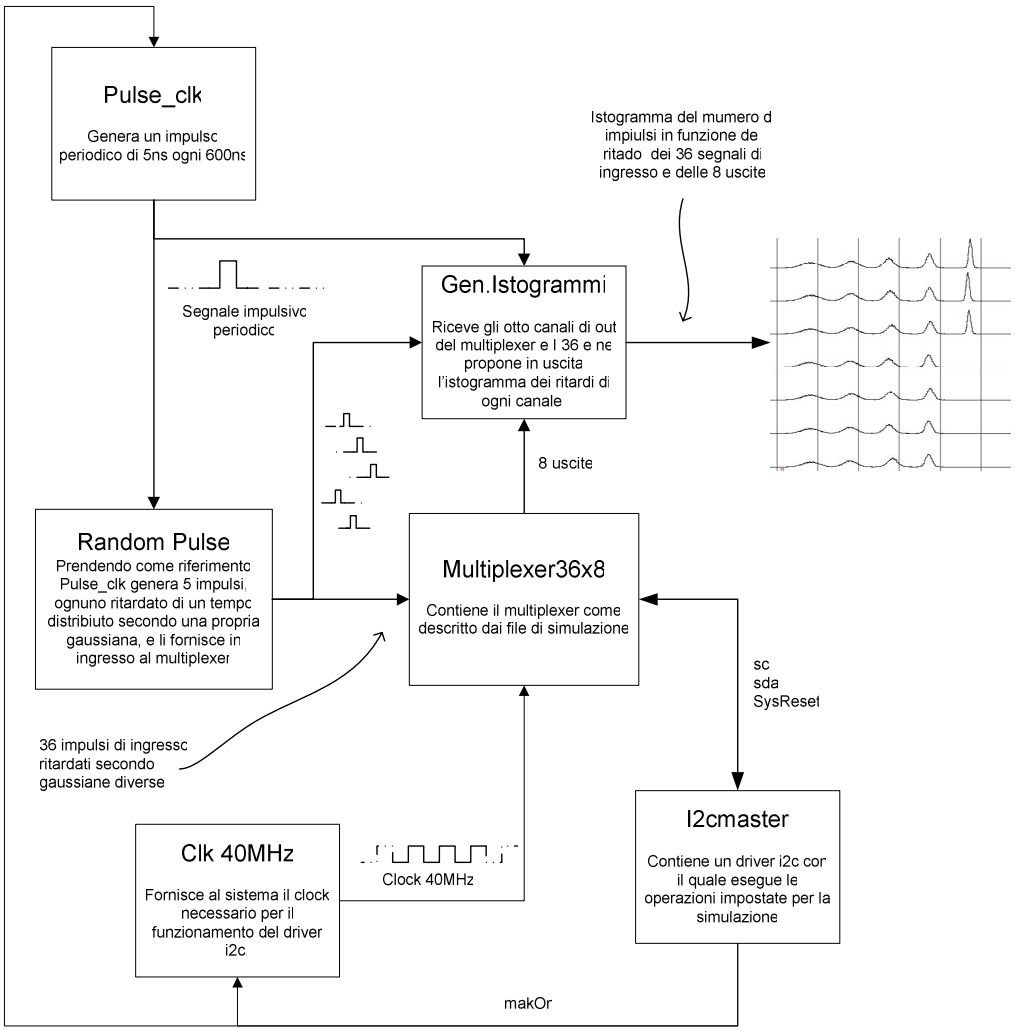


Figura 7.14: diagramma a blocchi del banco di test per la simulazione.

Per effettuare un test attinente allo scopo reale del multiplexer è stato simulato un sistema in cui gli ingressi sono costituiti dei segnali ritardati secondo delle distribuzioni gaussiane con valor medio e varianza diverse. Viene poi disegnato l'istogramma del ritardo subito da ogni impulso per ognuno dei canali di uscita aspettandosi che questo mostri la forma delle gaussiane, secondo cui sono distribuiti i segnali di ingresso a cui si riferisce, ma con valor medio incrementato del ritardo subito dal segnale.
Il banco di test è costituito dai seguenti moduli (figura 7.14):

1. **Pulse_clk (pulse_clk):** fornisce un impulso periodico, con periodo di 600ns e durata di 5ns, che costituisce il segnale di riferimento per la generazione degli impulsi di ingresso.

2. **Random Pulse (rnd_pulse_5):** utilizzando come riferimento il segnale impulsivo proveniente da Pulse_clk, produce in uscita 5 segnali impulsivi con la stessa forma ma ognuno ritardato di un tempo casuale estratto da una distribuzione gaussiana. Ogni uscita corrisponde una gaussiana diversa in modo da poterla riconoscere studiando l'andamento temporale degli impulsi.

3. **Multiplexer36x8 (MuxDelay36x8):** contiene il modello del multiplexer36x8 prodotto dal Xilinx ISE development system. I 36 canali di ingresso sono collegati ai cinque segnali uscenti da *Random Pulse* in modo che ognuno dei multiplexer 5x1 e 4x1, che idealmente lo compongono, riceva segnali di ingresso diversi tra loro. I cinque segnali del *Random Pulse* sono collegatio ai 36 ingressi secondo lo schema 5-5-5-5-4-4-4-4.

4. **I2C Master (i2c_master):** simula le funzionalità di un eventuale sistema di controllo. La sua funzione principale è quella di abilitare i registri di maschera scrivendo uno all'interno di essi tramite il

driver i2c-master contenuto al suo interno. Dispone di un ulteriore uscita, nominata msk_On, che segnala al banco di test quando l'operazione di scrittura nei registri è terminata, spengendo il generatore di clock e attivando quello di impulsi per semplificale l'algoritmo di simulazione e rendere più veloce il test.

5. **Clock 40 MHz :** fornisce il clock a 40 MHz necessario per il driver i2c del Multiplexer. Il clock è stato implementato direttamente nel file principale del banco di test (Test.v).

6. **Generatore di istogramma (isto):** è il modulo che genera l'istogramma del ritardo degli impulsi. In realtà ogni modulo serve un solo segnale di ingresso e possiamo pensare il generatore di istogrammi come l'insieme di tutti i moduli: uno per ogni uscita del multiplexer e uno per ognuna delle cinque uscite del modulo Random Pulse da usare come riferimento. Ogni modulo dispone di un vettore di 500 numeri interi che viene messo in relazione biunivoca con i 500 ns successivi alla generazione dell'impulso di Pulse_clk, preso come tempo di riferimento. Quando arriva l'impulso di ingresso viene calcolato il ritardo rispetto all'impulso di riferimento e incrementato il registro corrispondente. Contemporaneamente un'altra funzione legge i registri del vettore-istogramma uno alla volta, a distanza di 1 ns, e ne presenta in valore sui 32bit di uscita, graficando l'istogramma.

Utilizzando questo apparato sono stati studiati i canali di uscita in funzione del tempo. Tale test ha mostrato la validità del progetto mostrando istogrammi dei canali d'uscita del multiplexer uguali a quelli dei canali d'ingresso ma con valor medio incrementato del ritardo relativo al canale di provenienza.

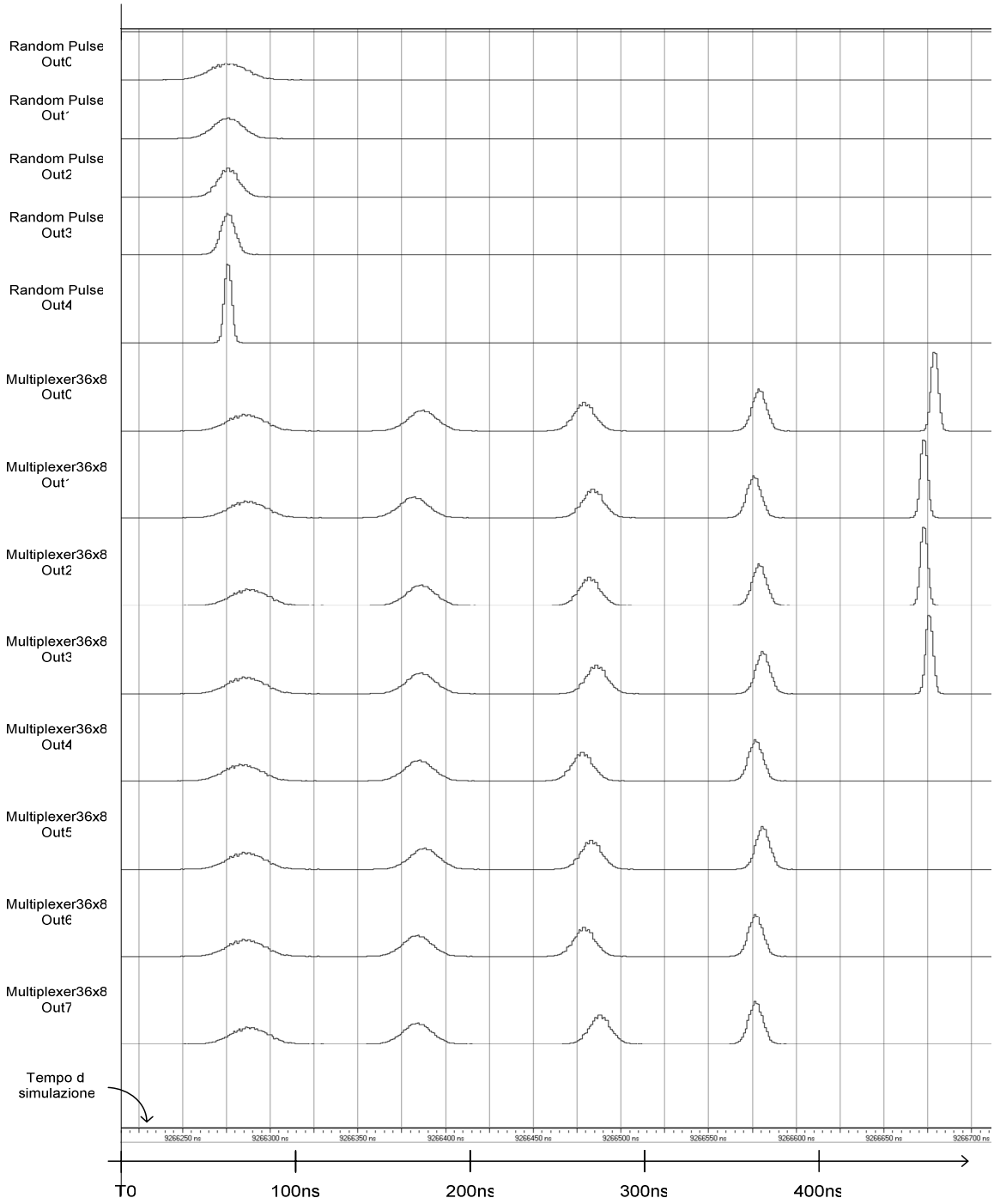


Figura 7.15: Risultato della simulazione del funzionamento del multiplexer ottenuto con il software ModelSim.

Appendice.

Il codice Verilog del Multiplexer 36x8.

A1 - Il multiplexer36x8.

```
// MuxDelay36x8
// Multipler 36x8 per la stazione di test
// Created:
// Francesco Iacoangeli 01-2003
//

`timescale 1 ns /100 ps

module MuxDelay36x8(in,
SYSRESET_,
scl,
sda,
i2c_addr_set,
clk,
out_p,out_n);

input clk;
input [35:0] in;
input [6:0] i2c_addr_set;
input SYSRESET_;
inout sda /* synthesis syn_noclockbuf = 1 */;
input scl /* synthesis syn_noclockbuf = 1 */;

output [7:0] out_p;
output [7:0] out_n;


reg [7:0] REG_mask0,REG_mask1,REG_mask2,REG_mask3,REG_mask4;
//registri dei bit di maschera
reg [7:0] i2cdata_in;


wire [7:0] i2csubaddr,i2cdata;
wire strobe /* synthesis syn_noclockbuf = 1 */;
wire write_,read_;
wire we_ = write_;
//wire [35:0] in_m;


/*****************************   I2C Mask ******************/
vb_i2c_slave_sync vb_i2c_slave_sync(
// i2c signals
.scl(scl),
```

```
.sda(sda),
.i2c_addr_set(i2c_addr_set),
// parallel out
.addr(i2csubaddr),
.data_out(i2cdata),
.data_in(i2cdata_in),
.strobe(strobe),
.write_(write_),
.read_(read_),
.clk(clk),
.SYSRESET_(SYSRESET_)) ;


/*****************************************************************
************/
//
// I2C Registers
// Management
//
`define ADDR_mask0      'h0
`define ADDR_mask1      'h1
`define ADDR_mask2      'h2
`define ADDR_mask3      'h3
`define ADDR_mask4      'h4

always @ (posedge strobe or negedge SYSRESET_)

begin
            if ( !SYSRESET_ )
            begin
                  REG_mask0 <= #5 0;
                  REG_mask1 <= #5 0;
                  REG_mask2 <= #5 0;
                        REG_mask3 <= #5 0;
                        REG_mask4 <= #5 0;
            end
      else
            begin
                  if ( SYSRESET_ )
                        begin
                                    if(!we_)
                                          begin

                                    case (i2csubaddr)
                                                `ADDR_mask0
:REG_mask0  <= #5  i2cdata[7:0];
                                                `ADDR_mask1:
REG_mask1  <= #5  i2cdata[7:0];
                                                `ADDR_mask2:
REG_mask2  <= #5  i2cdata[7:0];
                                                `ADDR_mask3:
REG_mask3  <= #5  i2cdata[7:0];
                                          `ADDR_mask4: REG_mask4
<= #5  i2cdata[7:0];

                                          default:REG_mask0 [7:0]
<= #5  i2cdata[7:0];
                                    endcase
```

```
                                        end  // !we
                        end
            end
end
//
// Read registers
//
always @ (i2csubaddr or REG_mask0 or REG_mask1 or REG_mask2
             or REG_mask3 or REG_mask4)

      begin
            case (i2csubaddr)
                  'h0: i2cdata_in[7:0]<= #5 REG_mask0;
                  'h1: i2cdata_in[7:0]<= #5 REG_mask1;
                        'h2: i2cdata_in[7:0]<= #5 REG_mask2;
                        'h3: i2cdata_in[7:0]<= #5 REG_mask3;
                  'h4: i2cdata_in[7:0]<= #5 REG_mask4;

            default:i2cdata_in[7:0]<= #5  8'hEB;
            endcase
      end


//-----------------------------------------------------------------
------------------

wire [35:0]
maskIn={REG_mask0,REG_mask1,REG_mask2,REG_mask3,REG_mask4};


/****************** And Mask **************/


MuxDelay4x
mux0(.MuxIn(in[35:32]),.MuxMask(maskIn[35:32]),.MuxOut_p(out_p[0])
,.MuxOut_n(out_n[0]));
MuxDelay4x
mux1(.MuxIn(in[31:28]),.MuxMask(maskIn[31:28]),.MuxOut_p(out_p[1])
,.MuxOut_n(out_n[1]));
MuxDelay4x
mux2(.MuxIn(in[27:24]),.MuxMask(maskIn[27:24]),.MuxOut_p(out_p[2])
,.MuxOut_n(out_n[2]));
MuxDelay4x
mux3(.MuxIn(in[23:20]),.MuxMask(maskIn[23:20]),.MuxOut_p(out_p[3])
,.MuxOut_n(out_n[3]));

MuxDelay5x
mux4(.MuxIn(in[19:15]),.MuxMask(maskIn[19:15]),.MuxOut_p(out_p[4])
,.MuxOut_n(out_n[4]));
MuxDelay5x
mux5(.MuxIn(in[14:10]),.MuxMask(maskIn[14:10]),.MuxOut_p(out_p[5])
,.MuxOut_n(out_n[5]));
MuxDelay5x
mux8(.MuxIn(in[9:5]),.MuxMask(maskIn[9:5]),.MuxOut_p(out_p[6]),.Mu
xOut_n(out_n[6]));
MuxDelay5x
mux6(.MuxIn(in[4:0]),.MuxMask(maskIn[4:0]),.MuxOut_p(out_p[7]),.Mu
xOut_n(out_n[7]));
```

```
endmodule
```

A2 - Il multiplexer4x1.

```
// MuxDelay4x
// Multipler 4x1 per la stazione di test
// Created:
// Francesco Iacoangeli 01-2003
//

module MuxDelay4x(MuxIn,MuxMask,MuxOut_p,MuxOut_n);
input [3:0] MuxIn;
output MuxOut_n;
output MuxOut_p;
input [3:0] MuxMask;

wire [3:0] in_dly;
wire [3:0] and_out;

wire myout0dly;


wire myout1dly0;

delay100 xlx0(.DelayIn(in_dly[1]),.DelayOut(myout1dly0));



wire myout2dly1;

delay100 xlx1(.DelayIn(in_dly[2]),.DelayOut(myout2dly0));
delay100 xlx2(.DelayIn(myout2dly0),.DelayOut(myout2dly1));


wire myout3dly2;

delay100 xlx3 (.DelayIn(in_dly[3]),.DelayOut(myout3dly0));
delay100 xlx4 (.DelayIn(myout3dly0),.DelayOut(myout3dly1));
delay100 xlx5 (.DelayIn(myout3dly1),.DelayOut(myout3dly2));



IBUF_LVDS XLXI_6 (.I(MuxIn[0]), .O(and_out[0]));
   /* synopsys attribute fpga_dont_touch "true"
   */
IBUF_LVDS XLXI_7 (.I(MuxIn[1]), .O(and_out[1]));
   /* synopsys attribute fpga_dont_touch "true"
   */
IBUF_LVDS XLXI_8 (.I(MuxIn[2]), .O(and_out[2]));
   /* synopsys attribute fpga_dont_touch "true"
   */
IBUF_LVDS XLXI_9 (.I(MuxIn[3]), .O(and_out[3]));
   /* synopsys attribute fpga_dont_touch "true"
   */
```

```
AND2 mux_0(.I0(and_out[0]),.I1(MuxMask[0]),.O(in_dly[0]));
AND2 mux_1(.I0(and_out[1]),.I1(MuxMask[1]),.O(in_dly[1]));
AND2 mux_2(.I0(and_out[2]),.I1(MuxMask[2]),.O(in_dly[2]));
AND2 mux_3(.I0(and_out[3]),.I1(MuxMask[3]),.O(in_dly[3]));



wire XLXN_11;

OR4 XLXI_12 (.I0(in_dly[0]), .I1(myout1dly0), .I2(myout2dly1),
.I3(myout3dly2),
      .O(XLXN_11));
   /* synopsys attribute fpga_dont_touch "true"
   */

//OBUF XLXI_10 (.I(XLXN_11), .O(MuxOut));
   /* synopsys attribute fpga_dont_touch "true"
   */


INV XLXI_16 (.I(XLXN_11), .O(XLXN_15));
   /* synopsys attribute fpga_dont_touch "true"
   */

OBUF_LVDS XLXI_17 (.I(XLXN_11), .O(MuxOut_p));
   /* synopsys attribute fpga_dont_touch "true"
   */

OBUF_LVDS XLXI_18 (.I(XLXN_15), .O(MuxOut_n));
   /* synopsys attribute fpga_dont_touch "true"
   */


endmodule
```

A3 - Il multiplexer5x1.

```
// MuxDelay5x
// Multipler 5x1 per la stazione di test
// Created:
// Francesco Iacoangeli 01-2003
//
module MuxDelay5x(MuxIn,MuxMask,MuxOut_p,MuxOut_n);
input [4:0] MuxIn;
output MuxOut_p;
output MuxOut_n;
input [4:0] MuxMask;

wire myout0dly;


wire [4:0] in_dly;
wire [4:0] and_out;
```

```
wire myout1dly0;

delay100 Xilinx0(.DelayIn(in_dly[1]),.DelayOut(myout1dly0));


wire myout2dly1;

delay100 Xilinx1(.DelayIn(in_dly[2]),.DelayOut(myout2dly0));
delay100 Xilinx2(.DelayIn(myout2dly0),.DelayOut(myout2dly1));


wire myout3dly2;

delay100 Xilinx3 (.DelayIn(in_dly[3]),.DelayOut(myout3dly0));
delay100 Xilinx4 (.DelayIn(myout3dly0),.DelayOut(myout3dly1));
delay100 Xilinx5 (.DelayIn(myout3dly1),.DelayOut(myout3dly2));


wire myout4dly3;

delay100 Xilinx6 (.DelayIn(in_dly[4]),.DelayOut(myout4dly0));
delay100 Xilinx7 (.DelayIn(myout4dly0),.DelayOut(myout4dly1));
delay100 Xilinx8 (.DelayIn(myout4dly1),.DelayOut(myout4dly2));
delay100 Xilinx9 (.DelayIn(myout4dly2),.DelayOut(myout4dly3));


wire XilinxN_17;



IBUF_LVDS Xilinx10 (.I(MuxIn[0]), .O(and_out[0]));
   /* synopsys attribute fpga_dont_touch "true"
   */
IBUF_LVDS Xilinx11 (.I(MuxIn[1]), .O(and_out[1]));
   /* synopsys attribute fpga_dont_touch "true"
   */
IBUF_LVDS Xilinx12 (.I(MuxIn[2]), .O(and_out[2]));
   /* synopsys attribute fpga_dont_touch "true"
   */
IBUF_LVDS Xilinx13 (.I(MuxIn[3]), .O(and_out[3]));
   /* synopsys attribute fpga_dont_touch "true"
   */
IBUF_LVDS Xilinx14 (.I(MuxIn[4]), .O(and_out[4]));
   /* synopsys attribute fpga_dont_touch "true"
   */


AND2 mux_0(.I0(and_out[0]),.I1(MuxMask[0]),.O(in_dly[0]));
AND2 mux_1(.I0(and_out[1]),.I1(MuxMask[1]),.O(in_dly[1]));
AND2 mux_2(.I0(and_out[2]),.I1(MuxMask[2]),.O(in_dly[2]));
AND2 mux_3(.I0(and_out[3]),.I1(MuxMask[3]),.O(in_dly[3]));
AND2 mux_4(.I0(and_out[4]),.I1(MuxMask[4]),.O(in_dly[4]));



OR5 Xilinx15 (.I0(in_dly[0]), .I1(myout1dly0), .I2(myout2dly1),
.I3(myout3dly2), .I4(myout4dly3),
      .O(XilinxN_17));
   /* synopsys attribute fpga_dont_touch "true"
```

```
   */

INV XLXI_4 (.I(XilinxN_17), .O(XLXN_18));
   /* synopsys attribute fpga_dont_touch "true"
   */

OBUF_LVDS XLXI_2 (.I(XilinxN_17), .O(MuxOut_p));
   /* synopsys attribute fpga_dont_touch "true"
   */

OBUF_LVDS XLXI_5 (.I(XLXN_18), .O(MuxOut_n));
   /* synopsys attribute fpga_dont_touch "true"
   */




endmodule
```

A4 - Il delay100.

```
//delay100
// Verilog model created from schematic delaytest100.sch - Thu Mar
06 13:51:39 2003
// Created:
// Francesco Iacoangeli 01-2003
//

`timescale 1ns / 1ps

module delay100(DelayIn, DelayOut);

input DelayIn;
output DelayOut;

wire DelayIn;
wire DelayOut;

delay xlnx0(.Del3In(DelayIn),.Del3Out(myout1));
delay xlnx1(.Del3In(myout1),.Del3Out(myout2));
delay xlnx2(.Del3In(myout2),.Del3Out(myout3));
delay xlnx3(.Del3In(myout3),.Del3Out(myout4));
delay xlnx6(.Del3In(myout4),.Del3Out(DelayOut));


//IBUF XLXI_6 (.I(DelayIn), .O(myout0));
   /* synopsys attribute fpga_dont_touch "true"
   */

//OBUF XLXI_7 (.I(myout7), .O(DelayOut));
   /* synopsys attribute fpga_dont_touch "true"
   */

endmodule
```

Il codice Verilog del banco di test.

B1 - Il banco di test.

```
`timescale 1 ns /100 ps
module test;
reg SYSRESET_;
reg clk;
wire mskOn;
tri1 scl,sda;
integer i;
wire [4:0] pulseTest;
wire pulse;

wire [0:7] out_p;
wire [0:7] out_n;

wire [31:0]
istogramma0,istogramma1,istogramma2,istogramma3,istogramma4,istogr
amma5

      ,istogramma6,istogramma7,istoIn0,istoIn1,istoIn2,istoIn3,ist
oIn4;

initial

begin
SYSRESET_ <= 1'b1; //porta SYSRESET a 1
#20;        //ci rimane per 20 unità di tempo,cioè 20ns
SYSRESET_ <= 1'b0; //lo porta a 0 e ci rimane per 200ns
#200;
SYSRESET_ <= 1'b1;//lo riporta a 1
#10000 $stop;
end



// istanzio i2c_master

i2c_master i2c_master (
  .scl(scl),
  .sda(sda),
  .SYSRESET_(SYSRESET_),
  .mskOn(mskOn)
  ) ;



  //

  glbl glbl ();


// istanzio MuxDelay36x8
```

```
wire [0:35] in={

      pulseTest[4],pulseTest[3],pulseTest[2],pulseTest[1],pulseTes
t[0],

      pulseTest[4],pulseTest[3],pulseTest[2],pulseTest[1],pulseTes
t[0],

      pulseTest[4],pulseTest[3],pulseTest[2],pulseTest[1],pulseTes
t[0],

      pulseTest[4],pulseTest[3],pulseTest[2],pulseTest[1],pulseTes
t[0],
            pulseTest[3],pulseTest[2],pulseTest[1],pulseTest[0],
            pulseTest[3],pulseTest[2],pulseTest[1],pulseTest[0],
            pulseTest[3],pulseTest[2],pulseTest[1],pulseTest[0],
            pulseTest[3],pulseTest[2],pulseTest[1],pulseTest[0]
            };



MuxDelay36x8 MuxDelay36x8 (.in(in[0:35]),
  .SYSRESET_(SYSRESET_),
  .scl(scl),
  .sda(sda),
  .i2c_addr_set(7'h72),
  .clk(clk),
  .out_p(out_p),.out_n(out_n));


 //clock 40MHz

  always
   begin : clk_gen
     clk <= 1'b0;
     #(25) ;
     clk <= 1'b1;
     #(25) ;
     end



 //istanzio il generatore di impulsi

pulse_clk pulse_clk (.pulseStart(mskOn),.pulse(pulse));



//istanzio il generatore di impulsi RANDOM

pulse5 pulse5 (.pulseIn(pulse),.pulseOut(pulseTest));



//istanzo gli istogrammatori per i canali di uscita

isto istoMux0
(.istoInput(out_p[0]),.istoStart(pulse),.istoOutput(istogramma0));
```

```
isto istoMux1
(.istoInput(out_p[1]),.istoStart(pulse),.istoOutput(istogramma1));
isto istoMux2
(.istoInput(out_p[2]),.istoStart(pulse),.istoOutput(istogramma2));
isto istoMux3
(.istoInput(out_p[3]),.istoStart(pulse),.istoOutput(istogramma3));
isto istoMux4
(.istoInput(out_p[4]),.istoStart(pulse),.istoOutput(istogramma4));
isto istoMux5
(.istoInput(out_p[5]),.istoStart(pulse),.istoOutput(istogramma5));
isto istoMux6
(.istoInput(out_p[6]),.istoStart(pulse),.istoOutput(istogramma6));
isto istoMux7
(.istoInput(out_p[7]),.istoStart(pulse),.istoOutput(istogramma7));



//istanzio gli istogrammatori per i canali di ingresso

isto isto0
(.istoInput(pulseTest[0]),.istoStart(pulse),.istoOutput(istoIn0));
isto isto1
(.istoInput(pulseTest[1]),.istoStart(pulse),.istoOutput(istoIn1));
isto isto2
(.istoInput(pulseTest[2]),.istoStart(pulse),.istoOutput(istoIn2));
isto isto3
(.istoInput(pulseTest[3]),.istoStart(pulse),.istoOutput(istoIn3));
isto isto4
(.istoInput(pulseTest[4]),.istoStart(pulse),.istoOutput(istoIn4));
//
//
// I2C Monitor (stampa su video le transizioni 12c

//
reg [7:0]count,dato;
reg first,read;
always @ (negedge sda)
begin
      if (scl)
            begin
                  $write ("\n",$time,"====> I2C_START <===\n");
                  count<=0;
                  first=1;
            end
end

//
// Stop
//
always @ (posedge sda)
begin
      if (scl)
            begin

                  $write ("\n",$time,"====> I2C_STOP <===\n");
                  count<=0;
            end
      end
//
```

```
// data
//
always @ (posedge scl)
begin
if (count==0) $write ($time,"====>");

$write ("%x=%x|",count,sda);
dato[7-count]=sda;
count=count+1;
if (count>8)
begin
count= 0;
if (first)
      begin

            if (!dato[0])
                  begin
                        $write (" WRITE@ADDR =%x",dato[7:1]);
                        read=0;
                  end


            else
                  begin
                        $write (" READ@ADDR =%x",dato[7:1]);
                        read=1;
                  end
      end
else
      begin
      if (read)  $write (" READ DATA=%x",dato);
       else $write (" WRITE DATA=%x",dato);
      end

if (!read | first)
      begin
            first=0;
            if (sda) $write (" NOACK\n");
            else $write (" ACK\n");
      end
else
      begin
            if (sda) $write (" LAST\n");
            else $write (" NEXT\n");
      end

end


end

endmodule
```

B2 - Il modulo i2c master.

```
// I2c Master
```

```
// Valerio Bocci
// valerio.bocci@roma1.infn.it
// June 2001
// i2c routine Code based from
// ELMB C source code
// Modified from
//Francesco Iacoangeli
//for test Multiplexer36x8
//april 2003
`timescale 1ns / 100ps
module i2c_master (scl,sda,SYSRESET_,mskOn) ;
output scl,mskOn ;
inout sda ;
input SYSRESET_;
reg sdaoe_,sda_out,scl,mskOn;
reg [7:0]  data_rd;
reg [7:0]  i;
assign sda= (!sdaoe_ & !sda_out)  ? 1'b0:'hz ;
reg capperi;


initial
begin
mskOn <= 1'b0;
scl<=1'b1;
sda_out<=1'b1;
sdaoe_<=1'b1;
#200;

//
// I2C Registers
//
//
`define ADDR_mask0      'h0
`define ADDR_mask1      'h1
`define ADDR_mask2      'h2
`define ADDR_mask3      'h3
`define ADDR_mask4      'h4

@(posedge SYSRESET_);
//
//
/*************** Writhe **************/
$display ("Scrivo tutti 1 nei registri di maschera \n");
i2c_start;
i2c_write({7'h72,1'b0});
i2c_write(`ADDR_mask0);// REG_mask0
i2c_write('hff);
i2c_stop;
//
i2c_start;
i2c_write({7'h72,1'b0});
i2c_write(`ADDR_mask1);// REG_mask1
i2c_write('hff);
i2c_stop;


i2c_start;
i2c_write({7'h72,1'b0});
```

```
i2c_write(`ADDR_mask2);// REG_mask2
i2c_write('hff);
i2c_stop;


i2c_start;
i2c_write({7'h72,1'b0});
i2c_write(`ADDR_mask3);// REG_mask3
i2c_write('hff);
i2c_stop;


i2c_start;
i2c_write({7'h72,1'b0});
i2c_write(`ADDR_mask4);// REG_mask4
i2c_write('hff);
i2c_stop;
#1000;
$display ("Fine i2c \n");


$stop;
 mskOn <= 1'b1;
end
/*******END initial*******/
task i2c_start;
begin
  /* Generate I2C START condition */
$write ($time," i2c START \n");
  sdaoe_<=0;
  #250;
  sda_out<=1'b1;
  #250;
  scl<=1'b1;
i2c_delay;
  sda_out<=0;
i2c_delay;
  scl<=0;
i2c_delay;
  end

endtask

/* -------------------------------------------------------------
--------- */

task i2c_stop;

begin
$write ($time," i2c STOP \n");
  /* Generate I2C STOP condition */
  scl<=1'b1;
i2c_delay;
   sdaoe_<=0;
   #250;
  sda_out<=0;

i2c_delay;
  sda_out<=1'b1;
```

```
  i2c_delay;

     sdaoe_<=1'b1;
       #250;
end

endtask


task i2c_read ;
output [7:0] byt;
input last;
reg [5:0] i;
reg [7:0] byt;
begin

    byt<=0;

  /* Set SDA to input, without pull-up */
#250;
sdaoe_<= 1'b1;
#250;
  sda_out<= 1'b0;
#250;
  /* Receive BYTE, MSB first */
  for( i=0; i<8; i=i+1 )
    begin

      scl <= 1'b1;
#250;
      i2c_delay;
#250;
      if( !scl ) $write ($time," Error: I2C SCL STUCK LOW \n");
#250;

      /* Get ready for next bit */
      byt[7:0] <= {byt[6:0],1'b0};
#250;


      if( sda ) byt<=#500 byt+1;
#250;

      scl <= 1'b0;
#250;
      i2c_delay;
#250;
    end

  /* Get ready for acknowledge: is it the last byte to read ? */
  if( last )
  begin
    sda_out<= 1'b1;
#250;
end
  else
  begin
    sda_out<= 1'b0;
#250;
```

```
end

  /* Set SDA to output */
  sdaoe_<= 1'b0;
#250;
  scl <= 1'b1;
#250;
  i2c_delay;
#250;
  scl <= 1'b0;
#250;
  i2c_delay;
#250;
$write ($time,"I2c READ %x   \n",byt);
end
endtask
/* -----------------------------------------------------------------
--------- */
task i2c_write;
input [7:0] byt;
reg [5:0] i;
begin

$write ($time," i2c WRITE %x\n",byt);
  /* Write a byte to the I2C bus  */

  /* Set SDA back to output */
  sdaoe_<=1'b0;
#250;

  /* Send BYTE, MSB first */
  for( i=0; i<8; i=i+1)
    begin
      if( byt[7])
begin
      sda_out<=1'b1;
#250;
end
else

begin
      sda_out<=1'b0;
#250;
end

      /* Get ready for next bit */
      byt[7:0]<={byt[6:0],1'b1};
#250;

      scl<=1'b1;
#250;
      i2c_delay;
#250;
      if( !scl ) $write ($time," Error:I2C_SCL_STUCK_LOW\n");
      scl<=1'b0;
#250;
      i2c_delay;
#250;
    end
```

```
  /* To receive the acknowledge, set SDA to input (without pull-
up) */
  sdaoe_<=1'b1;
#250;
  sda_out<=1'b0;
#250;

  /* Get the acknowledge */
  scl<=1'b1;
#250;
  i2c_delay;
#250;
  if( sda ) begin
  $write ($time,"Error: I2C_ACK_MISSING\n");
  capperi=1;
  end
  scl<=1'b0;
#250;
  i2c_delay;
#250;

  /* Set SDA back to output */
  sdaoe_<=1'b0;
#250;
end
endtask
/* ------------------------------------------------------------------
--------- */
task i2c_delay;
begin
      #10000;
end
 endtask

 endmodule
```

B3 - Il generatore di impulsi.

```
// Pulse.v
// generatore di impulsi
//
// Created:
// Francesco Iacoangeli 04-2003
//


`timescale 1ns / 100ps
module pulse_clk (pulseStart,pulse) ;

input pulseStart;
output pulse;
reg pulse;
integer i;
initial
```

```
begin
pulse = 0;
end
initial @ (posedge pulseStart)
begin
      for (i=0 ; i<10000; i=i+1)
      begin : clk_gen
         pulse <= 1'b0;
         #(600);
         pulse <= 1'b1;
         #(5);
      end
end
endmodule
```

B4 - Il generatore di impulsi RANDOM.

```
// pulse5
// generatore di 5 impulsi RANDOM secondo gaussiane diverse
//
// Created:
// Francesco Iacoangeli 04-2003
//

`timescale 1ns / 100ps


module pulse5 (pulseIn,pulseOut) ;
input pulseIn;
output [0:4] pulseOut;
reg [0:4] pulseOut;
reg [0:7] i;
real gauss0,gauss1,gauss2,gauss3,gauss4;
real seed0,seed1,seed2,seed3,seed4;
initial
begin
pulseOut=0;
seed0=239;
seed1=234;
seed2=634;
seed3=743;
seed4=916;
end

always @(posedge pulseIn)
      begin : random_pulse

   gauss0 = $dist_normal(seed0,60,2);
   gauss1 = $dist_normal(seed1,60,4);
   gauss2 = $dist_normal(seed2,60,6);
   gauss3 = $dist_normal(seed3,60,8);
   gauss4 = $dist_normal(seed4,60,10);
```

```
      pulseOut[0] <= #(gauss0) 1'b1;
      pulseOut[0] <= #(5+gauss0) 1'b0;

      pulseOut[1] <= #(gauss1) 1'b1;
      pulseOut[1] <= #(5+gauss1) 1'b0;

      pulseOut[2] <= #(gauss2) 1'b1;
      pulseOut[2] <= #(5+gauss2) 1'b0;

      pulseOut[3] <= #(gauss3) 1'b1;
      pulseOut[3] <= #(5+gauss3) 1'b0;

      pulseOut[4] <= #(gauss4) 1'b1;
      pulseOut[4] <= #(5+gauss4) 1'b0;
      end

endmodule
```

B5 - Il generatore di istogramma.

```
// isto.v
// generatore di istogramma
//
// Created:
// Francesco Iacoangeli 04-2003
//

`timescale 1 ns /100 ps
module isto(istoInput,istoStart,istoOutput);

input istoInput,istoStart;
output [31:0] istoOutput;
reg [31:0] istoOutput;

time timeOut;
time time0;

integer j;
integer i;
integer isto [0:500] ;


initial
begin
istoOutput = 0;
for (j=0;j<=500;j=j+1)
      begin
      isto[j] = 0;
      end
end


always @(posedge istoStart) time0=$time;
```

```
always @(posedge istoInput)
      begin
      timeOut=$time;
      j=(timeOut-time0);
      isto[j]=isto[j]+1;
      end

always @(posedge istoStart)
      begin
      for (i=0;i<=500;i=i+1)
            begin
            #1;
            istoOutput=isto[i];
            end
      end
endmodule
```

Il “pin-out” della Spartan-II XCV300E.

Nome del Segnale	Nome del pin	N° del pin	Direzione	Standard di I/O
in<0>	CS_L28P_YY	B20	INPUT	LVDS
in<0>_n	WRITE_L28N_YY	A20	INPUT	LVDS
in<1>		B19	INPUT	LVDS
in<1>_n		A19	INPUT	LVDS
in<2>	VREF	B18	INPUT	LVDS
in<2>_n	VREF	A18	INPUT	LVDS
in<3>	L25P_YY	B17	INPUT	LVDS
in<3>_n	L25N_YY	A17	INPUT	LVDS
in<4>		B16	INPUT	LVDS
in<4>_n		A16	INPUT	LVDS
in<5>		B15	INPUT	LVDS
in<5>_n		A15	INPUT	LVDS
in<6>		B14	INPUT	LVDS
in<6>_n		A14	INPUT	LVDS
in<7>	VREF	B13	INPUT	LVDS
in<7>_n	VREF	A13	INPUT	LVDS
in<8>		A10	INPUT	LVDS
in<8>_n		B10	INPUT	LVDS
in<9>	L7P_Y	A8	INPUT	LVDS
in<9>_n	L7N_Y	B8	INPUT	LVDS
in<10>	L3P_YY	A6	INPUT	LVDS
in<10>_n		B6	INPUT	LVDS
in<11>	L1P_Y	B4	INPUT	LVDS
in<11>_n		C5	INPUT	LVDS
in<12>		D2	INPUT	LVDS
in<12>_n		D1	INPUT	LVDS
in<13>	VREF	E2	INPUT	LVDS
in<13>_n	VREF	E3	INPUT	LVDS
in<14>	VREF	F2	INPUT	LVDS
in<14>_n	VREF	F1	INPUT	LVDS
in<15>		G2	INPUT	LVDS
in<15>_n		G1	INPUT	LVDS
in<16>	L112P_YY	H2	INPUT	LVDS
in<16>_n	L112N_YY	H1	INPUT	LVDS
in<17>		K1	INPUT	LVDS
in<17>_n		L1	INPUT	LVDS
in<18>		P1	INPUT	LVDS
in<18>_n		P2	INPUT	LVDS

Nome del Segnale	Nome del pin	N° del pin	Direzione	Standard di I/O
in<19>		T1	INPUT	LVDS
in<19>_n		U1	INPUT	LVDS
in<20>		W1	INPUT	LVDS
in<20>_n		V2	INPUT	LVDS
in<21>		AB3	INPUT	LVDS
in<21>_n		AA3	INPUT	LVDS
in<22>		AB6	INPUT	LVDS
in<22>_n		AB5	INPUT	LVDS
in<23>	L85P_YY	AA7	INPUT	LVDS
in<23>_n		AB7	INPUT	LVDS
in<24>		AA8	INPUT	LVDS
in<24>_n		AB8	INPUT	LVDS
in<25>		AA9	INPUT	LVDS
in<25>_n		AB9	INPUT	LVDS
in<26>		AA13	INPUT	LVDS
in<26>_n		AB13	INPUT	LVDS
in<27>		Y15	INPUT	LVDS
in<27>_n		AA15	INPUT	LVDS
in<28>	VREF	Y17	INPUT	LVDS
in<28>_n	VREF L64N_YY	AA17	INPUT	LVDS
in<29>	L60P_YY	AA20	INPUT	LVDS
in<29>_n	L60N_YY	AB20	INPUT	LVDS
in<30>	D7_L59P_YY	Y22	INPUT	LVDS
in<30>_n	INIT_L59N_YY	W21	INPUT	LVDS
in<31>		U22	INPUT	LVDS
in<31>_n		V22	INPUT	LVDS
in<32>		T22	INPUT	LVDS
in<32>_n		T21	INPUT	LVDS
in<33>		R22	INPUT	LVDS
in<33>_n		R21	INPUT	LVDS
in<34>	L51P_YY	P21	INPUT	LVDS
in<34>_n		P22	INPUT	LVDS
in<35>		N21	INPUT	LVDS
in<35>_n		N22	INPUT	LVDS
out_p<0>		K21	OUTPUT	LVTTL
out_p<1>		J21	OUTPUT	LVTTL
out_p<2>		H22	OUTPUT	LVTTL
out_p<3>		G22	OUTPUT	LVTTL
out_p<4>		F22	OUTPUT	LVTTL
out_p<5>		E21	OUTPUT	LVTTL
out_p<6>	VREF	D21	OUTPUT	LVTTL
out_p<7>	DOUT_BUSY_L29P_YY	C21	OUTPUT	LVTTL
i2c_addr_set<0>		P20	INPUT	LVTTL
i2c_addr_set<1>	D5_L51N_YY	P22	INPUT	LVTTL
i2c_addr_set<2>		P18	INPUT	LVTTL
i2c_addr_set<3>		P19	INPUT	LVTTL

Nome del Segnale	Nome del pin	N° del pin	Direzione	Standard di I/O
i2c_addr_set<4>		N22	INPUT	LVTTL
i2c_addr_set<5>		D20	INPUT	LVTTL
i2c_addr_set<6>	DIN_D0_L29N_YY	C22	INPUT	LVTTL
SYSRESET_		AB21	INPUT	LVTTL
clk		AB12	INPUT	LVTTL
scl	L61P	Y19	INPUT	LVTTL
sda	L61N	AA19	BIDIR	LVTTL

Il protocollo I2c.

Il protocollo i2c di trasmissione dati fa uso di sole 2 linee di trasmissione denominate SCL e SDA. La configurazione trasmettitore-ricevitore è di tipo "Master-Slave" in cui il primo ha il compito di inviare il segnale "strobe" sulla line SCL per segnalare l'invio di dati sull'altra linea. La linea SDA contiene i dati e può essere comandata da entrambi i dispositivi per permettere la trasmissione in entrambi le direzioni. Alla fine di ogni serie di dati trasmessi viene generato un "acknowledge" dal dispositivo master in caso di successo. I dati vengono trasmessi otto bit alla volta e ogni pacchetto inizia con un segnale di START, seguito dai dati con i rispettivi acknowledge, e finisce con un segnale di STOP. La lettura del bit trasmesso avviene quando SCL è alto mentre le permutazioni devono avvenire quando è basso.

I segnali di START e di STOP sono generati dal dispositivo master: lo START è generato da una transizione da livello alto a livello basso della linea SDA mentre SCL è alto mentre lo STOP da una transizione inversa di SDA sempre con il segnale SCL alto.Nel caso in cui, alla fine di un trasferimento di un pacchetto, si voglia trasmetterne subito un altro, è possibile generare un nuovo START al posto dello STOP e iniziare la trasmissione.

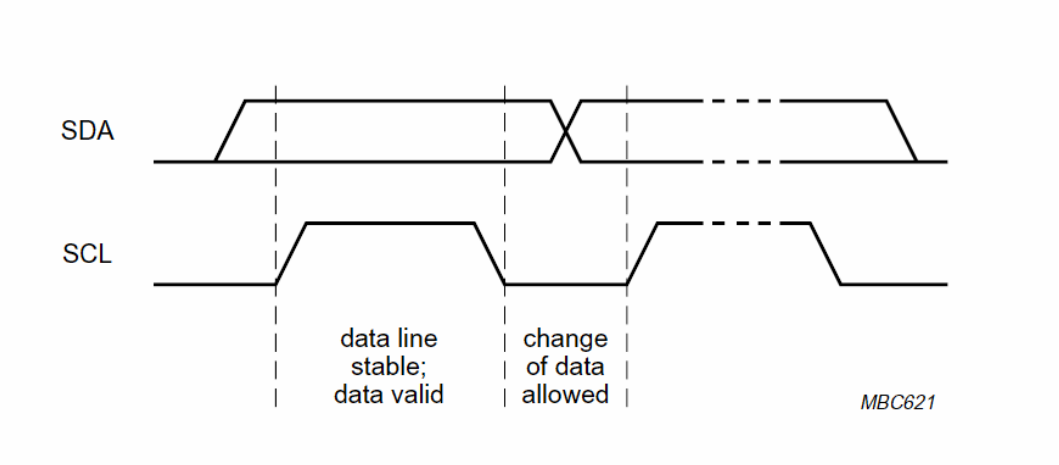


Figura D.1: Trasferimento di un bit con il protocollo I2C.

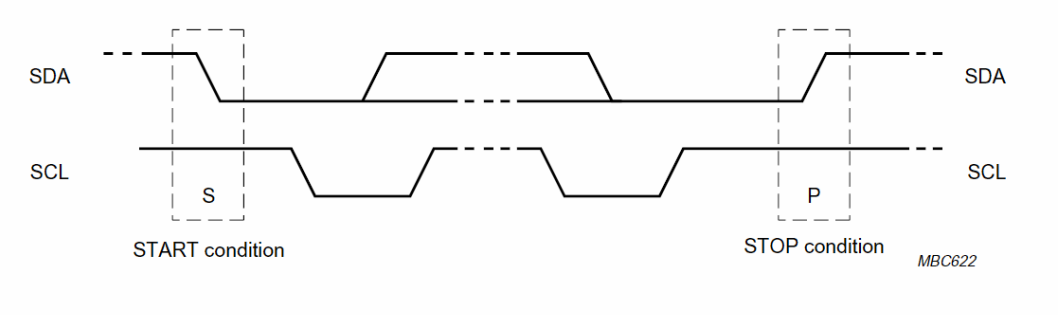


Figura D.2: Condizioni di START e di STOP di una transizione I2C.

Il segnale di acknowledge viene generato dal dispositivo master che effettua una transizione sulla linea SCL, durante la quale il dispositivo ricevitore (che può essere anche lui stesso) mantiene SDA al livello basso. Se il dispositivo non ha ricevuto bene il segnale deve mantenere SDA alto al momento dell' acknowledge generando così uno stato di "not- acknowledge" che provoca l'arresto della trasmissione da parte del master con un segnale di STOP. Questo evento viene utilizzato anche per terminare la trasmissione di dati dal dispositivo slave a quello master: arrivati all'ultimo dato in master non trasmette l'acknowledge, lo slave interpreta il comando rilasciando l'SDA in modo che possa essere generato il segnale di STOP.

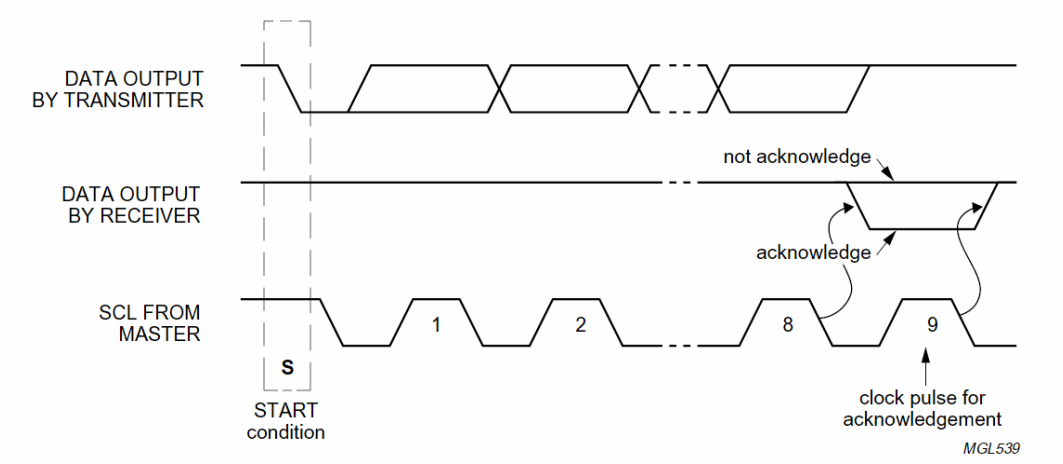


Figura D.3: Generazione dell'acknowledge in una transizione I2C.

L'indirizzamento dei dati e la scrittura e la lettura dei registri avvengono secondo le specifiche mostrate in figura D.5 e D.6.

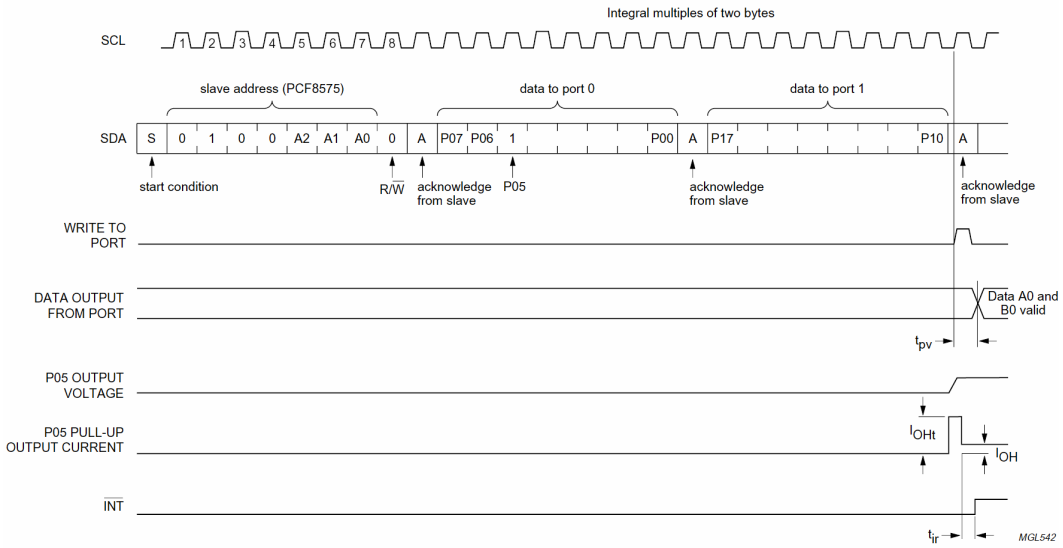


Figura D.5: Procedimento di scrittura con protocollo I2C.

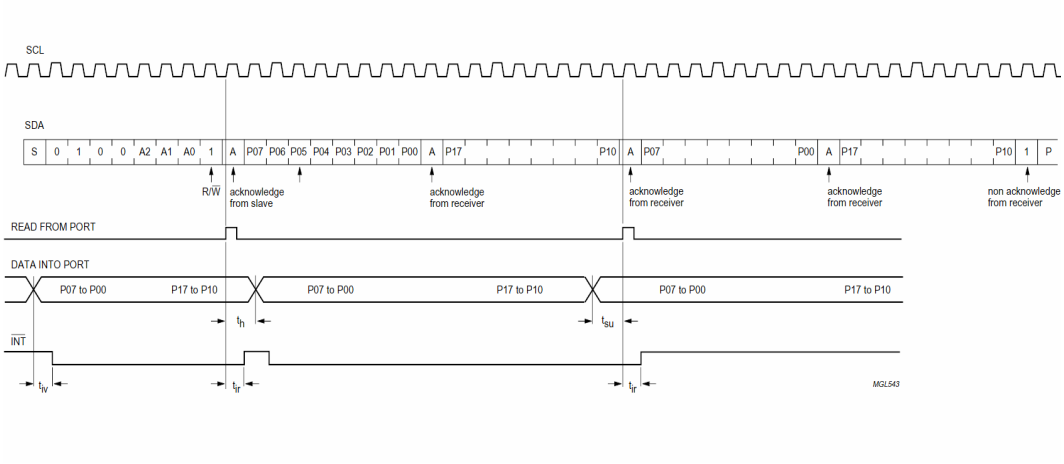


Figura D.6: Procedimento di lettura con protocollo I2C.

Esempi di programmazione del firmware e del client.

A causa della lunghezza dei listati non è possibile riportare per intero il lavoro fatto sul firmware e sul software client.
Di seguito sono riportati parti dei listati a titolo di esempio.
In particolare gli esempi, per quanto riguarda il firmware, si riferiscono agli oggetti 4600, 4601 e 4700 dell'SDO e alla routine di scrittura nei registri dell'Actel contenuti rispettivamente nei file sdo.c e i2c.c.
Per il software client si è scelto di riportare il menù di selezione delle funzionalità dell'Actel e le funzioni per l'utilizzo degli oggetti CAN per la scrittura e la lettura nei suoi registri.

E1 - *Oggetto Power Control (4500).*

```
 //--------------------START WRITE OBJ---------------------------

case OD_PWRCTRL_HI:

    switch (od_index_lo)
        {
        case (OD_PWRCTRL):

            if ((od_subind <=3)  & chkpsw())
                {
                    if (msg_data[4]==0xff)
                        {
                        elmb_read_i2c(3,0x21,0x2,I2C_DATA);
                        I2C_DATA[0]=I2C_DATA[0] | (0x01<<od_subind);
                        elmb_write_i2c(3,0x21,0x2,I2C_DATA);
                        }
                    else
                        {
                        elmb_read_i2c(3,0x21,0x2,I2C_DATA);
                        I2C_DATA[0]=I2C_DATA[0] & ~(0x01<<od_subind);
                        elmb_write_i2c(3,0x21,0x2,I2C_DATA);
                        }
                }// if (od_subind <=3)

            else
                {
                /* The sub-index does not exist */
                sdo_error = SDO_ECODE_ATTRIBUTE;
                }// else if od_subind<=3

        break;// case (OD_PWRCTRL):


        case (OD_PWRCTRLPSW):

            switch( od_subind )
                {
                    case 0:
                    set_pwrpsw(msg_data);
                    break;

                    default:
                    /* The sub-index does not exist */
                    sdo_error = SDO_ECODE_ATTRIBUTE;

                }//switch od_subind

        break;// case (OD_PWRCTRLPSW):

        default:

        /* The index can not be accessed, does not exist */
        sdo_error = SDO_ECODE_NONEXISTENT;

        break;

    }//switch od_index_lo

break;//case OD_PWRCTRL_HI

//-------------------------END WRITE OBJ----------------------
```

E2 - Oggetto per la lettura della password per il Power Control (4501).

```
//----------------------START READ OBJ-----------------------------

    case OD_PWRCTRL_HI:

         switch (od_index_lo)

                {

                case (OD_PWRCTRL):

                               if ((od_subind <=3))
                                    {
                                        elmb read i2c(3,0x21,0x2,I2C DATA);
                                        I2C DATA[0]=I2C_DATA[0] & (0x01<<
                                        od_subind);
                                        if (I2C_DATA[0]!=0) msg_data[4]=
                                        0xff;
                                         else msg data[4]=0x00;
                                     }// if (od_subind <=3)

                                else
                                    {
                                    /* The sub-index does not exist */
                                    sdo_error = SDO ECODE ATTRIBUTE;
                                    }// else if od_subind<=3

                break;// case (OD_PWRCTRL):


                case (OD_PWRCTRLPSW):

                    switch( od_subind )
                        {
                            case 0:
                            show pwrpsw(msg_data);
                            break;

                            default:

                            /* The sub-index does not exist */

                            sdo_error = SDO_ECODE_ATTRIBUTE;

                        }//switch od_subind

                break;// case (OD_PWRCTRLPSW):


                default:

                /* The index can not be accessed, does not exist */

                sdo_error = SDO_ECODE_NONEXISTENT;

                break;


                }//switch od_index_lo

    break;//case OD PWRCTRL HI
//--------------------------END OBJ--------------------------------------
```

E3 - *Oggetto per la scrittura nei registri dell'Actel (4700).*

```
//--------------------------START WRITE OBJ-------------------------
case OD_ACTEL_HI:

    switch (od_index_lo)
        {

            case (OD ACTEL):
               if (od_subind <=13)
                    {
                    I2C_DATA[0]=od_subind;

                    for (i=0;i<=3;i++)
                        {
                        I2C_DATA[i+1]=msg_data[i+4];
                        }

                    elmb write i2c(3,0x72,0x2,I2C_DATA);
                    }// if (od_subind <=3)

                else
                    {
                    /* The sub-index does not exist */
                    sdo_error = SDO ECODE ATTRIBUTE;
                    }// else if od_subind<=3

            break;// case (OD_ACTEL):

            default:

            /* The index can not be accessed, does not exist */
            sdo_error = SDO_ECODE_NONEXISTENT;

            break;

        }//switch od_index_lo

break;//case OD_ACTEL_HI

//--------------------------END WRITE OBJ--------------------------
```

E4 - Routine I2C per la scrittura nei registri dell'Actel.

```
// Routines i2c in local mode VB & FI apr 2003
// addr in 7 bits format

BYTE actel_read_i2c(int channel,BYTE addr,BYTE subaddr,BYTE len,BYTE *data)

{

     BYTE i=0;

    i2c_start(channel);

    i2c_write(channel, (addr << 1) & 0xFE);

    i2c_write(channel,subaddr);

    i2c_stop(channel);

    i2c_start(channel);

    i2c_write(channel,((addr<<1) | 0x01) );

        for (i=0;i<len;i++)
          {
             *(data+i) =i2c_read(channel,i==len-1);
          }

    i2c_stop(channel);

    /* Return the status byte and reset it to zero */

  return 0;

}
```

E5 - *Menu “Test_Pulse_Select” del software client.*

```
//************************* TEST PULSE SEL MENU ************************************

void Test_pulse_menu (BOOL version)
{
    char *test_pulse_str[]=
{
"int_pulse",
"clk40  ",
"BC_pulse",
"reg_pulse"
    };

char *int_pulse_str[]=
{
"div_pulse" ,
"dly_pulse" ,
"sw_pulse " ,
"",
"async_start_stop" ,
"sync_start_stop" ,
"sw_start_stop",
""
    };
BYTE tmis_h,tmis_l;
int tmis;
BYTE dato;

Actel_read_reg(ADDR_TEST_PULSE_SEL,&test_pulse);
Actel_read_reg(ADDR_INT_PULSE_SEL,&int_pulse);

if (!version) system("cls");
version=0;
                                printf(
                                "*********************************************************\n"
                                );
    if (test_pulse!=INT_PULSE)  printf( "\t\t%s_menu\t\n",test_pulse_str[test_pulse] );
    if (test_pulse==INT_PULSE)  printf( "\t\t%s_menu\t\n",int_pulse_str[int_pulse] );
                                printf(
                                "*********************************************************\n"
                                );
    if ((test_pulse<=3))    printf( "t (test_pulse_source) [%s]                  \t*\n",
    test_pulse_str[test_pulse]);
    if ((test_pulse==INT_PULSE))
        {
        printf( "i (int_pulse_source) [%d %s]              \t*\n",int_pulse,int_pulse_str[
        int_pulse]);
        }
    if (test_pulse==REG_PULSE)
        {
         Actel_read_reg(ADDR_PULSE,&pulse);
         printf( "0 reg_Pulse=0                                    \t*\n" );
         printf( "1 reg_Pulse=1                                     \t*\n");
         printf( "b Toggle (reg_pulse=%x)                           \t*\n",pulse );
        }
    if ((test_pulse==INT_PULSE) &( (int_pulse==DIV_PULSE) | (int_pulse==ASYNC_START_STOP)))
        {
        Actel_read_reg(ADDR_TDIV,&dato);
        printf( "d (reg_TDIV set)[%i ns (%d)]                      \t*\n",dato*25,dato );
        }
    if ((test_pulse==INT_PULSE) & ((int_pulse==DLY_PULSE)|(int_pulse==SYNC_START_STOP)))
        {
        Actel_read_reg(ADDR_TDLY,&dato);
        printf( "r reg_TDLY set(%i ns)                             \t*\n",dato*25 );
        }
    if ((test_pulse==INT_PULSE) & ((int_pulse==ASYNC_START_STOP)|(int_pulse==SYNC_START_STOP)|
    (int_pulse==SW_START_STOP)))
        {
        Actel_read_reg(ADDR_TMIS_H,&tmis_h);
        Actel_read_reg(ADDR_TMIS_L,&tmis_l);
        tmis=(tmis_h*256+tmis_l);
        printf( "m reg_TMIS set(%d ns)                               \t*\n",tmis*25 );
        }
    if ((test_pulse==INT_PULSE) & ((int_pulse==SW_PULSE)|(int_pulse==SW_START_STOP)))
        {
        printf( "g Generate Pulse                                  \t*\n");
        }
        printf( "x read/write reg                                 \t*\n");
        printf( "*********************************************************\n" );

return 0;
}
//*************************END TEST PULSE SEL MENU ************************************
```

E6 - Funzioni di acesso agli oggetti CANOpen per la scrittura nei registri dell'Actel (Actel.c).

```
#include <windows.h>
#include "canopen.h"
#include "ActelReg.h"

int Actel_read_reg(BYTE i2caddr,BYTE *data)
{
WORD index;
BYTE can_msg_cnf[4];
BYTE can_msg_rtn[4];
WORD len;
BYTE subidx;
index=0x4700;
subidx=i2caddr;

COP_ReadSDO (CanInterface,NodeID,1, index,subidx,&len,can_msg_rtn,
can_msg_cnf);
*data=can_msg_rtn[0];
return 0;
}
//*****************************START Actel_write_reg ******************


int Actel_write_reg(BYTE i2caddr,BYTE data)
{
WORD index;
BYTE message[4];
BYTE can_msg_cnf[4];
BYTE subidx=i2caddr;
index=0x4700;
message[0]=data;
COP_WriteSDO (CanInterface,  NodeID,1, index, subidx, 4, message,
can_msg_cnf);
return 0;
}
//*****************************END TEST PULSE SEL MENU ******************
```

Glossario

Acknowledge
Risposta positiva alla fine della trasmissione di un dato. Nel protocollo I2C l'acknoweledge consiste nel mantenere l'SDA basso mentre il master effettua una transizione dell'SCL.

ALICE
A Large Ion Collider Experiment.

ATLAS
A Toroidal LHC Apparatus.

Bootloader
Parte del firmware in cui sono contenute le informazioni di avvio e inizializzazione del sistema.

BC_pulse
Linea sulla quale viene trasmesso un segnale impulsivo periodico generato all'interno dei moduli PDM dividendo il clock di LHC ed in fase con quest'ultimo.

Brunch crossing
Vedi BX.

BX
Bunch Crossing. Indica l'intervallo temporale tra ogni collisione dei pacchetti di protoni, ed è pari a 25ns.

BX-Identifier
Bunch Crossing Identifier. Numero che identifica in quale BC del fascio è avvenuto un evento. La loro numerazione va da 0 a 3563 per ogni orbita in LHC.

CAN
Controller Area Network. Protocollo di trasmissione dati su bus.

CMOS
Complementary MOSfet. Famiglia logica che fa uso di transistor ad effetto di campo. La loro carattestica è la ridottissima potenza dissipata in condizioni statiche: circa 10 nW per porta logica e un ampio intervallo di valori di tensione di alimentazione: da 3V a 15V.

CMS
Compact Muon Solenoid

DAQ
Data Acquisition system.

Derandomizer
Memoria che immagazzina i dati per un massimo uguale al tempo necessario per il loro trigger in modo da inviarli al read-out con un rate costante.

DIALOG
Diagnostic, time Adjustment and LOGics.

DLL

Delay Locked Loop. Indica un circuito integrato a reazione negativa che ha la funzione di mantenere costante il ritardo di un segnale di ingresso rispetto al clock interno.

ECS

Experiment Control System. È il sistema di controllo dell'apparato comprendente l'elettronica di controllo dei rivelatori e del sistema di trigger.

ELMB

Embedded Local Monitor Board.Scheda dotata di un microcontrollore ATmega128 AVR su cui si basa la Service Board.

FPGA

Field Programmable Gate Array. Circuito integrato la cui logica è programmabile.

FE

Front End. Indica la generica l'interfaccia elettronica che converte i segnali analogici del rivelatore in segnali adatti all'elettronica di trigger.

FIFO

First In First Out. Indica un tipo di memoria nella quale i dati che entrano per primi escono per primi.

Hex

Indica un numero in formato esadecimale.

I2C

Protocollo di comunicazione, vedi appendice D.

I2c-like LVDS.

Protocollo di comunicazione, nello standard LVDS, sviluppato per l'ECS che si distingue dal protocollo I2C per la presenza di due linee SDA, una d'ingresso e una d'uscita.

IB

Intermediate-Board.

Interupt

Evento che richiede attenzione immediata inviato alla CPU.

LEP

Large Electron-Positron Collider. Collisore di elettroni-positroni.

LHC

Large Hadron Collider.

LHCb

Large Hadron Collider bPhysics, rivelatore di LHC.

Long i2c branches

Rami I2C della lunghezza massima di 10 m su cui possono essere collegati fino a 256 periferiche I2C.

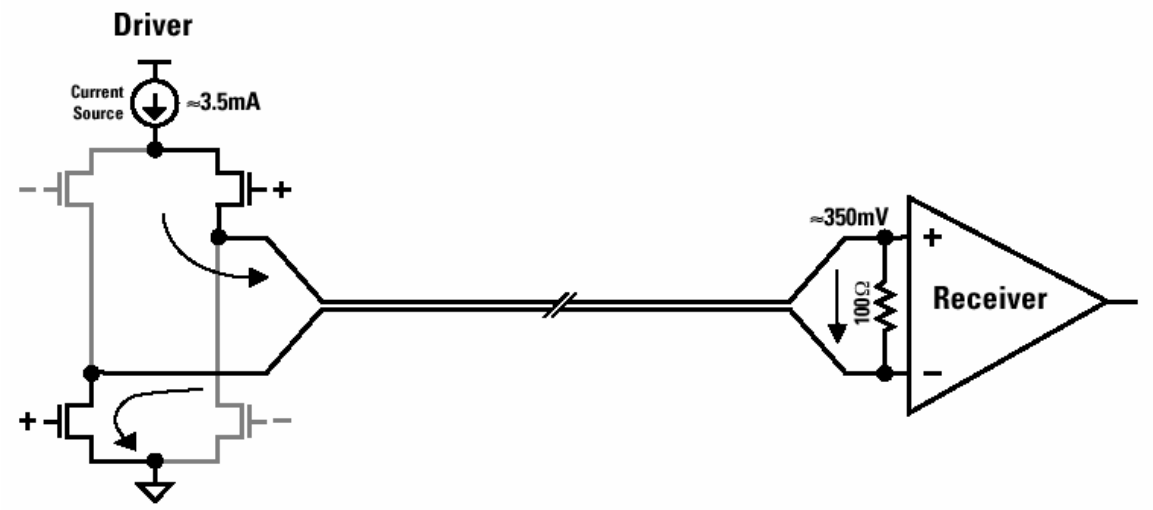
LSB

Least Significant Bit. Cifra meno significativa in un numero binario

LVDS

Low Voltage Differential Signaling. E' un sistema di comunicazione differenziale basato cioè sulla differenza di potenziale tra due canali di trasmissione. E' un sistema studiato per dare un'alta velocità di trasmissione, dell'ordine dei Mbps, con una bassa dissipazione di potenza e poco rumore. Nella figura seguente vi è lo schema generale di funzionamento. Si vede come il driver sia sostanzialmente una

sorgente di corrente di 3.5mA che guida la linea. Il ricevitore ha una alta impedenza in continua, così la maggior parte della corrente va sulla resistenza di 100Ω ad una tensione di 350 mV. Quando il driver commuta cambia la direzione della corrente nella resistenza creando il corrispondente stato logico 0 o 1. Il vantaggio rispetto agli schemi ad un solo canale è soprattutto sulla bassa dipendenza dal rumore.



Schema LVDS.

Mappatura
Operazione con la quale il software di implementazione di una FPGA trascrive la logica nelle celle.

MSB
Most Significant Bit. Cifra più significativa in un numero binario

Network
Serie di computer connessi tra loro in modo da potersi scambiare dati .

Not-acknowledge
Mancata risposta adel dispositivo ricevente alla richiesta di acknowledge.

ODE
Off Detector Electronics.

PAD
Zona del rivelatore di muoni identificata da un canale logico o dall'incrocio di canali logici provenienti da due strips.

PC-CAN-interface
Iterfaccia CAN tra sistema e PC esterno.

PDM
Pulse Distributor Module

Piazzamento
Operazione con cui si distribuisce nella maniera ottimale la logica nell'FPGA

Power cycling

Procedura che spegne una ELMB portando la tensione di alimentazione sotto il livello di funzionamento e la riaccende dopo un tempo abbastanza lungo da poter trascurare gli effetti capacitivi del dispositivo

RAM

Random Access Memory. Memoria ad accesso casuale.

Sbroglio

Processo con cui si determinano i collegamenti fra i vari gate delle celle.

SEE/SEU

Single Event Effects/Upset. Effetto indotto dale radiazioni ionizzanti che fanno commutare in maniera anomala lo stato di un bit di un circuito.

Stack

Pila di dati incui vengono salvati I dati necessari al ripristino di un applicazione quando viene effettuato un salto.

Strip

Zona del rivelatore di muoni contenente più di una pad su un unica linea trasversale o longitudinale.

Subroutine

Insieme di istruzioni che permettono ad un computer di svolgere una specifica operazione.

Tagging

Identificazione del sapore di una particella.

TDC

Time to Digital Converter. Indica un circuito che è in grado di misurare I tempi di segnali digitali.

Test_pulse

Linea lungo la quale vengono trasmessi I segnali di test dell'ECS generati nella Service Board.

Toggle

Operazione di negazione di un bit.

Tracker

Hardware e software che permettono lo studio della traiettoria di una particella.

Tracking.

Studio della traiettoria di una particella.

Trigger

Operazione di identificazione delle particelle con le caratteristiche volute.

TCF

Timing and Fast Control system

TTC-RX

Chip recivers su cui è basato il modulo PDM e il sistema TCF.

Verilog HDL

HDL è l'acronimo di Hardware Description Language. Il Verilog HDL è un linguaggio per la descrizione e la simulazione di sistemi digitali (hardware).

VME

Versa Module Europa. E' un sistema di comunicazione molto versatile ed utilizzato in numerosi settori. La sua nascita risale al 1981 ed è dovuta a diverse compagnie elettroniche ed il suo standard è stato definito dal IEEE 1014-1987,

che è quello a cui facciamo riferimento visto che attualmente ne esistono diversi. Le sue caratteristiche principali sono:

- Architettura MASTER/SLAVE.
- Bus Asincrono, quindi non sono usati clock per coordinare il trasferimento dati.
- Velocità di trasferimento dati variabile, che può raggiungere una banda passante di 40 Mbit/s.
- Sistema degli indirizzi a 16 o 32 bit.
- Profondità delle parole tra gli 8 ed i 32 bit.
- Capacità di multiprocessazione.
- Presenza di un protocollo di interrupt di 7 livelli.
- Numero di schede che può contenere un crate: 21.

Per poter gestire configurazioni a più master, su ciascun crate è installata una scheda "controller" che gestisce l'arbitrato tra le altre schede. Al suo interno vi sono 4 tipi di bus: quello per il trasferimento dati, per il trasferimento dati con arbitrato, il bus degli interrupt, quello di utilità.

Referenze

[1] The Atlas Experiment.
http://pdg.lbl.gov/atlas/atlas.html

[2] The CMS Experiment.
http://www.phys.ufl.edu/hee/cms/

[3] The Alice Experiment.
http://map.web.cern.ch/ALICE/welcome.html

[4] G. Passaleva, "Violazionedi CP nei mesoni B" , 5/6/2001

[5] D.E. Groom et al., The European Physical Journal **C15,** 1 (200)
http://pdg.lbl.gov/

[6] W.A.Ponce, "Penguin Diagrams In The Kobayashi-Maskawa Model", Print-79-0650 (MASS.U., AMHERST).

[7] Marta Calvi, "Physics reach LHCb".
Commissione Scientifica Nazionale I,Frascati 4 Febbraio 2003

[8] LHCb Technical Proposal CERN 1998,
"A Large Hadron Collider Beauty Experiment for Precision Measurements of CP Violation and Rare Decays",
http://lhcb.cern.ch/technicalproposal/

[9] The Vertex Detector Home Page.

http://lhcb-vd.web.cern.ch/lhcb-vd/Default.htm

[10] RICH detectors Home Page.

http://lhcb-rich.web.cern.ch/lhcb-rich/

[11] I.Azhgirey et al., "Development of MARS Code Package for Radiation Problems Solution of Electro-Nuclear Istallation Design", in: Proc. of XV Conference on Charged Particles Accelerators, Protvino, October 22-24 1996.

[12] C.Zeitnitz and T.A.Gabriel, The Geant-Calor interface User's Guide (1999)
http://www.physik.uni-mainz.de/zeitnitz/gcalor/gicalor.html

[13] R. Le Gac et al., "Particle fluxes in the LHCb Muon System Comparisons of GCALOR and MARS calculation", LHCb 1999-036 Muon.

[14] A. Lai, “Muon architecture review”,
CERN 27-3-2003

[15] A.Lai et al., LHCb note 2000-50

[16] B.Bochin et al., “Wire Pad Chamber for the LHCb Muon System”,
LHCb 2003

[17] G.Bencivenni et al, “A triple-GEM detector with pad readout for the inner region of the first
LHCb muon station”
http://www.lnf.infn.it/esperimenti/lhcb/gem/pub/lhcb-2001-051.pdf

[18] A. Lai, A. Sciubba, V. Bocci, G. Martelletti,“Muon Detector Front-end Architecture: an update”;
Reference: LHCb 2001-030;

[19] S. Cadeddu and A. Lai, “DIALOG-β DATA SHEET”,
I.N.F.N. Sezione di Cagliari, Cagliari, Italy
Reference: LHCb 2003-016 MUON

[20]. A.Lai – S. Cadeddu , “DIagnostics time Adjustment and LOGics”,
CERN 27/3/03

[21] A. Balla, P. Ciambrone, G. Felici, “ODE Board L0 front-end for muon system”.

[22]. ELMB technical documentation
http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/ELMB/SB/index.html

[23] V.Bocci,G.Chiodi, F. Iacoangeli,W.Rinaldi , “Data sheet Service Board”,
INFN Roma March 2003.

[24] Data sheet ATmega128.
http://www.atmel.com/dyn/resources/prod_documents/2467s.pdf

[25]
http://www.nikhef.nl/pub/departments/ct/po/html/ELMB128/ELMB128resources.html

[26].DataSheet PCF8575 Remote 16-bit I/O expander for I2C-bus.Feb1999

http://agenda.cern.ch/askArchive.php?base=agenda&categ=a03650&id=a03650s1t7%2Ftransparencies%2FODE_Overview.ppt

[27] J. Christiansen, A. Marchioro, P. Moreira* and T., “J. Christiansen, A. Marchioro, P. Moreira* and T.”.
http://ttc.web.cern.ch/TTC/TTCrx_manual3.8.pdf

[28] R. Jacobsson / CERN, B. Jost / CERN, Z. Guzik, / Warsaw Soltan Institute for Nuclear Studies, “TFC Switch Specifications”, LHCb Technical Note LHCb Technical Note.
Reference: LHCb DAQ 2001-018

[29] NIKHEF, “CANopen high-level protocol for CAN-bus”, Amsterdam March 20, 2000. Version 3.0.
http://www.nikhef.nl/pub/departments/ct/po/doc/CANopen30.pdf

[30] Davide Pinci and Walter Rinaldi, “The Roma1 test tools for MWPC”
http://agenda.cern.ch/askArchive.php?base=agenda&categ=a031003&id=a031003s2t2%2Fmoreinfo%2FCERN_May2003.pdf

[31] Stephen Brown and Jonathan Rose, Department of Electrical and Computer Engineering
University of Toronto, “Architecture of FPGAs and CPLDs: A Tutorial”.
http://www.eecg.toronto.edu/~brown/survey.html

[32] Xilinx, “Spartan-IIE Complete Data Sheet”
http://direct.xilinx.com/bvdocs/publications/ds077.pdf

[33] Walter Rinaldi, “CABLAGGIO STRUMENTAZIONE STAZIONE DI TEST PER CAMERE A FILI MUON SYSTEM LHCb”

[34] Rajeev Madhavan, “Quick Reference for Verilog® HDL”
http://parmita.com/verilogfaq/files/ref.pdf

[35] Xilinx, “Spartan-IIE 1.8V FPGA Family: Pinout Tables”
http://direct.xilinx.com/bvdocs/publications/ds077_4.pdf

Ringraziamenti.

Desidero ringraziare i miei relatori Giovanni Vittorio Pallottino, per la disponibilità e la collaborazione durante la stesura di questa Tesi, Valerio Bocci, per la fiducia accordatami, gli insegnamenti e l'aiuto ricevuti durante tutto il periodo di lavoro e Giuseppe Martelletti per la fiducia e gli aiuti ricevuti.

Vorrei inoltre ringraziare:

Annette Frenkel e Gianni Penso per l'aiuto nella stesura dei capitoli sulla stazione di test.

Silvano Di Marco e Antonio Rossi per la disponibilità e la preziosa assistenza tecnica nella fase di test della Service Board.

Giacomo Chiodi e Walter Rinaldi per il tempo dedicatomi e i numerosi aiuti teorici e pratici nella misure di jitter.

Cristian Della Polla per l'aiuto nella stesura e nella correzione di questo documento.

Un caloroso ringraziamento va a tutti i componenti del laboratorio di elettronica della sezione di Roma dell'INFN per i numerosi aiuti ricevuti , in particolare:

Felice Cidronelli, , Francesca Pastore, Francesco Pulcinella, , Riccardo Vari, Stefano Veneziano.

Infine desidero ringraziare Barbara Bellagamba per l'aiuto morale e l'incoraggiamento nei momenti di maggior impegno.