Università degli Studi di Lecce
Facoltà di Ingegneria

Corso di Laurea in Ingegneria dell'Informazione

Tesi di Laurea

SVILUPPO DI UN OSCILLATORE SINCRONIZZATO COL GPS PER APPLICAZIONI DI TIME STAMPING

Relatore:
Chiar.mo Prof. Marco Panareo

Laureando:
Andrea Donno

Anno Accademico 2005/2006

INDICE

CAPITOLO 4

SVILUPPO DEL FIRMWARE DELL'FPGA

INTRODUZIONE

I Raggi Cosmici sono particelle e nuclei atomici di alta energia che, muovendosi quasi alla velocità della luce, colpiscono la terra da ogni direzione. L'esistenza dei Raggi Cosmici fu scoperta dal fisico tedesco Victor Hess agli inizi del ventesimo secolo. All'epoca gli scienziati si trovavano di fronte a un problema che non riuscivano a spiegare: sembrava che nell'ambiente ci fosse molta più radiazione di quella che poteva essere prodotta dalla radioattività naturale.

Nel 1912, Hess decise di tentare un esperimento per risolvere la questione ancora aperta. Egli caricò su un pallone aerostatico un dispositivo per misurare le particelle cariche detto elettroscopio a foglie e intraprese un viaggio che dimostrò come la quantità di particelle cariche (e quindi di radiazione) aumentava con l'altitudine. Questo significava che la radiazione sconosciuta non aveva origine terrestre (come la radioattività naturale) ma proveniva dallo spazio esterno, da cui il nome di Raggi Cosmici.

A partire da questo primo esperimento i raggi cosmici sono stati intensamente studiati e adesso sappiamo molte cose sul loro conto. Da misure fatte su palloni aerostatici a grande altitudine o su satelliti sappiamo che la grandissima maggioranza dei raggi cosmici e' costituita da protoni (circa 90%). Abbiamo poi nuclei atomici (ovvero atomi privi dei loro elettroni) di svariati elementi, da quelli più leggeri come l'elio (circa 9%) fino ai più pesanti (circa 1%) come ferro e addirittura uranio.

Si pensa che i Raggi Cosmici, almeno quelli con energie fino a 10^{15} eV, vengano accelerati in seguito alle esplosioni di Supernovae nella nostra Galassia. Una esplosione di Supernova produce una fortissima onda d'urto che si propaga nel gas interstellare ed e' in grado di accelerare le particelle e i nuclei anche ad energie molto elevate come quelle che vediamo nei raggi cosmici.

Quando i Raggi Cosmici entrano nell'atmosfera terrestre collidono con i nuclei di cui essa e' composta. In queste collisioni viene prodotto un gran numero di particelle che a loro volta interagiscono o decadono creandone delle altre. Il risultato e' quello che viene chiamato "shower" (Fig. 1), o sciame di particelle.
Per studiare i raggi cosmici di energia elevata si usano esperimenti situati sulla superficie terrestre. Questi esperimenti rivelano i raggi cosmici secondari prodotti nell'interazione del primario con l'atmosfera. Dalle caratteristiche dello "shower" di particelle si può ricavare l'energia e la direzione del raggio cosmico primario.

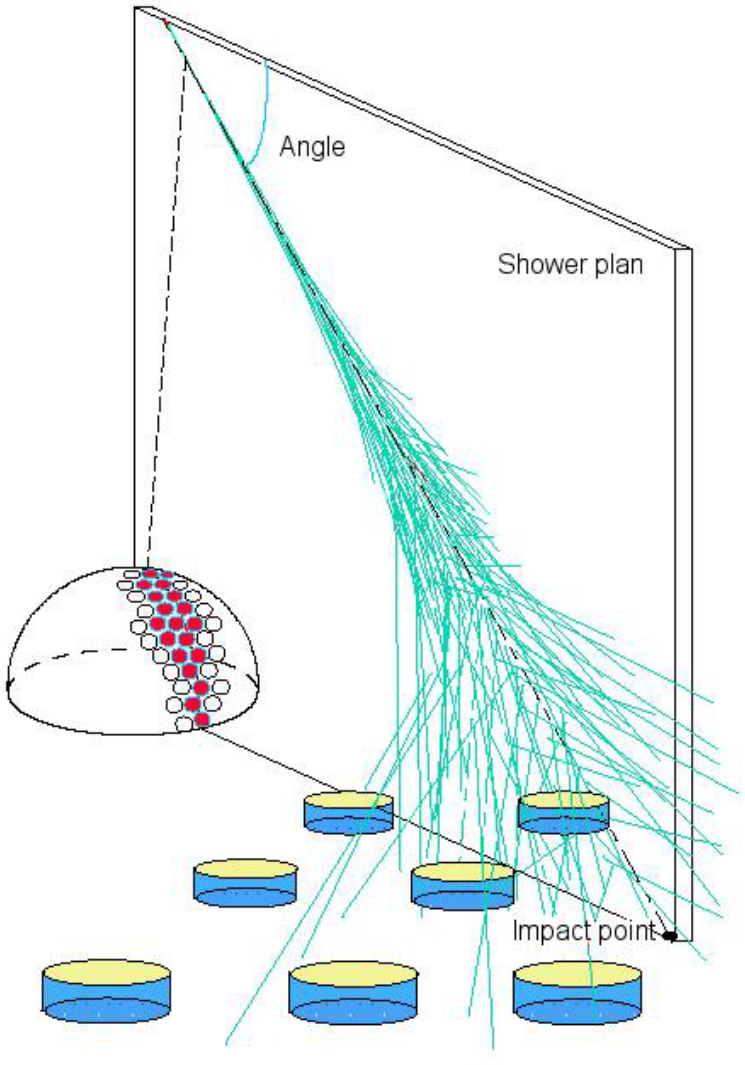


Fig. 1 – Angolo di incidenza del raggio cosmico primario e direzione delle particelle generate dall'interazione con l'atmosfera.

Questo e' lo scopo del progetto EEE (Extreme energy events) che prevede l'installazione di una serie di dispositivi, per la rivelazione dei raggi cosmici, chiamati MRPC (Multigap Resistive Plate Chamber) in diverse sedi scolastiche che vanno dall'estremo Nord al limite Sud Italia.

Uno dei problemi che ci si pone nell'affrontare l'esperimento, che corrisponde anche all'obbiettivo di questa tesi, è dovuto al fatto che come detto in precedenza abbiamo uno sciame di particelle che per essere rivelate hanno bisogno di dispositivi distribuiti su di un'ampia estensione territoriale, così i dati estrapolati dai vari rivelatori non possono essere controllati da un unico terminale e soprattutto da un unico clock, quindi non si può ottenere facilmente una correlazione temporale tra un dato e l'altro.

Per questo si impiega come riferimento temporale il sistema GPS che fa uso di orologi atomici ad alta precisione per fornire indicazioni estremamente accurate. Il ricevitore GPS riceve ogni secondo un segnale chiamato pulse per second (PPS) sincronizzato con l'UTC (Universal Coordinated Time) che è oggi il sistema di riferimento per la misura del tempo accettato in tutto il mondo, per questa ragione l'esperimento EEE prevede l'uso di una stazione GPS locale per ogni rivelatore MRPC.

Nell'intervallo di un secondo però possiamo avere più di un evento rivelato da parte di un MRPC, per questo si ha bisogno di una sincronizzazione tra il ricevitore GPS ed un oscillatore al quarzo che nel nostro caso lavora a 50 Mhz in modo da ottenere dei riferimenti di tempo molto precisi (dell'ordine dei nanosecondi).

Grazie alla presenza del GPS che ci fornisce il tempo assoluto, sincronizzato con il quarzo siamo in grado di effettuare un'applicazione di time stamping che consente di registrare, insieme alla misura, le informazioni sul tempo, cioè ricreare in modo accurato l'evoluzione temporale dei dati rivelati registrando il valore di un contatore, opportunamente sincronizzato, ogni volta che si verifica un evento nel rivelatore.

Questa tesi è stata svolta con il supporto del Centro Studi e Ricerche *Enrico Fermi* di Roma e della sezione di Lecce dell'Istituto Nazionale di Fisica Nucleare.

Capitolo 1

Il Sistema satellitare GPS

1.1. Cenni Storici

Il sistema NAVSTAR – GPS (NAVigation System for Timing and Ranging – Global Positioning System), più comunemente indicato con l'acronimo GPS, avviato dagli U.S.A. a partire dagli anni '70, e completato nel 1994, è stato realizzato per motivi principalmente militari, per rispondere all'esigenza del Ministero della difesa degli Stati Uniti di seguire il percorso di mezzi militari sulla terraferma ed in mare in modo da localizzarne la posizione in ogni momento e consentirne eventuali operazioni di supporto e di salvataggio.

Nei primi anni '60 il primo esperimento a prendere vita è il TRANSIT, frutto della collaborazione tra alcune delle più importanti organizzazioni governative, i cui contributi maggiori sono da ricercarsi nel campo degli algoritmi di predizione della posizione dei satelliti. Nel 1964 prende vita un altro precursore del GPS, chiamato TIMATION, nel quale si ritrovano le prime applicazioni di orologi atomici a bordo dei satelliti. Da qui in poi seguono diversi sistemi come il System 621B ed il SECOR.

Allo scopo di unificare i contributi di ognuno dei precedenti in un unico sistema, si istituisce nel 1968 un comitato congiunto chiamato NAVSEG (NAVigation Satellite Executive Group), con il compito di mettere a punto tutte le specifiche, al fine di dar vita cinque anni dopo all' esperimento chiamato GPS.

Tale programma entra nel vivo a partire dal 1969, quando la Rockwell International produce i primi satelliti GPS; il primo lancio avviene nel luglio del 1974, seguito dopo quattro anni dal lancio del primo di undici satelliti appartenenti al blocco che nel 1985 renderà il sistema funzionante. Nel febbraio 1989 iniziano i lanci dei 28 satelliti dei quali 24 operativi e 4 riservati ad eventuali sostituzioni, che renderanno operativo nel 1994 il sistema GPS.

Fino ad oggi non sono state apportate sostanziali modifiche escluso la rimozione di alcuni disturbi ed il lancio dei satelliti di nuova generazione chiamati BLK IIA (Fig. 1.1) e BLK IIR.

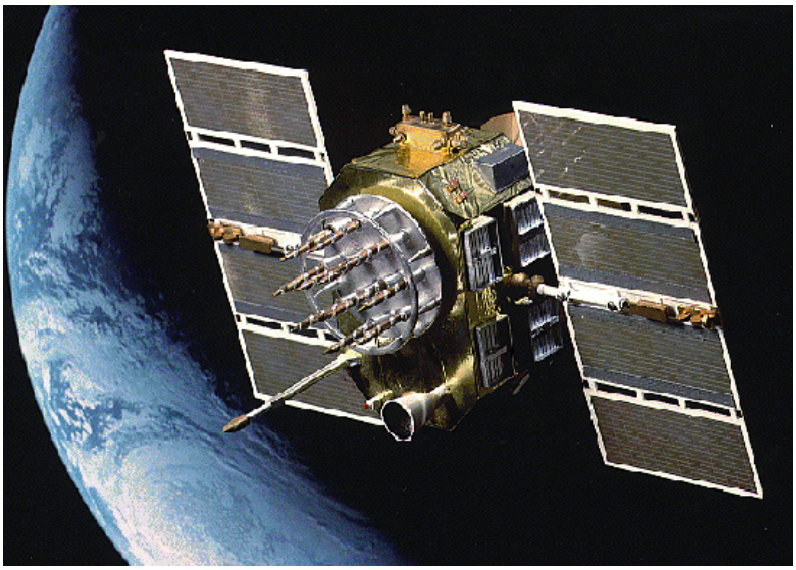


Fig. 1. 1 – Satellite BLK IIA

1.2. Configurazione del sistema GPS

Il GPS è un sistema di individuazione della posizione che utilizza 24 satelliti artificiali, divisi in gruppi di quattro che ruotano attorno alla terra alla quota di circa 20.200 Km in orbite distanti fra loro di un angolo di 60° e formanti un angolo di 55° rispetto al piano equatoriale. Esso consente la misura della posizione dell'utente nelle tre coordinate spaziali (latitudine, longitudine e altitudine) e del tempo UTC (Universal coordinated time) con una copertura globale e continua. Inoltre è possibile ricavare altre grandezze, se richieste, quali velocità, direzione, accelerazione ed altro ancora. Ovviamente la precisione sarà
limitata da tutta una serie di sorgenti di errori inevitabili (e che verranno descritte in seguito), ma soprattutto dal tipo di ricevitore utilizzato la cui qualità influisce sulle prestazioni complessive.

Sono previste due classi di utenza: gli utenti militari, che fruiscono del Precise Positioning Service (PPS), e gli utenti civili, che sfruttano lo Standard Positioning Service (SPS). La differenza in termini di prestazioni tra PPS e SPS è ottenuta artificialmente tramite i meccanismi di crittografia Selective Availability (SA) e di Anti-Spoofing (AS). Il primo consiste nell'introduzione voluta di errori aggiuntivi sui parametri di navigazione (rimossa dal DOD dal maggio 2000, ma con riserva di riattivazione in caso di necessità senza preavviso). Questo tipo di disturbo serve per negare una piena accuratezza all'utente SPS, mentre non ha effetti su utenti PPS in quanto possiedono gli elementi per eliminare questa crittografia del segnale. Il secondo è attivato per evitare tentativi di jamming da parte di utenti non autorizzati, cioè serve a far in modo che nessun possa inviare repliche del codice proveniente dai satelliti con l'intento di ingannare il ricevitore. La configurazione complessiva del sistema GPS comprende tre distinti segmenti:

- Il segmento SPAZIALE, formato dalla costellazione satellitare GPS orbitante intorno alla Terra;
- Il segmento di CONTROLLO, costituito da cinque stazioni GPS permanenti poste lungo l'Equatore;
- Il segmento UTENTE, rappresentato da chiunque sia dotato di un ricevitore GPS.

Il segmento spaziale è costituito da una costellazione di 24 satelliti operativi, come già accennato, affiancati da 4 satelliti previsti per eventuali sostituzioni. I satelliti sono disposti, a gruppi di 4 o 5, su sei orbite centrate attorno alla Terra, in modo che siano visibili da qualsiasi punto della superficie terrestre e in qualsiasi momento almeno 4 satelliti al di sopra di un angolo di elevazione rispetto all'orizzonte di 15°.
Le orbite, di tipo ellittico, sono equispaziate tra loro di 60° e presentano un angolo di inclinazione di 55° rispetto all'equatore ed un raggio approssimativo di 26,560 km (Fig. 1.2).
I satelliti non sono di tipo geostazionario, ma hanno un tempo di rivoluzione attorno alla Terra pari alla metà del giorno siderale, equivalente a circa 11 ore e 58 minuti, con una velocità di 3874 m/sec.

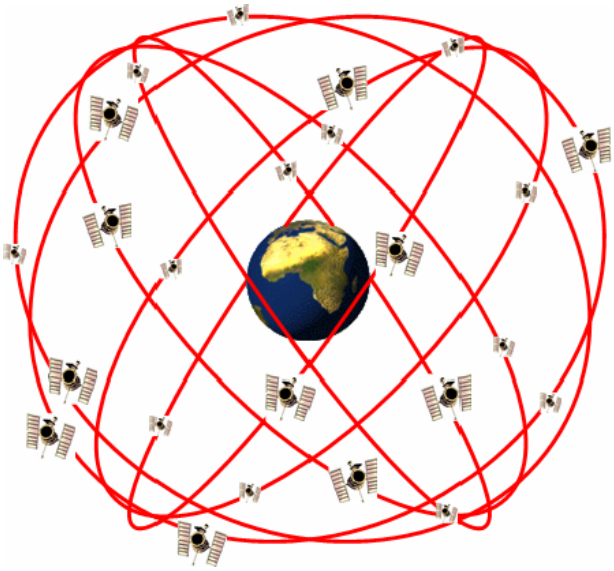


Fig. 1.2 – Costellazione delle orbite dei satelliti GPS

Componenti fondamentali di ciascun satellite sono i quattro orologi atomici di bordo, due al Cesio e due al Rubidio, che garantiscono un errore inferiore al secondo per un periodo che va da 30,000 ad un milione di anni e che servono per la generazione dei segnali in trasmissione; tali orologi, infatti, danno luogo ad un oscillatore con una frequenza base di 10.23 MHz, da cui è possibile ricavare tutte le frequenze in gioco.

Per rendere il più indipendente possibile la potenza ricevuta dal punto di ricezione, il diagramma di radiazione dei trasmettitori presenta un profilo sagomato opportunamente sull'angolo di apertura; tale distribuzione deve tenere conto anche del non uniforme guadagno di antenna del ricevitore di terra rispetto all'angolo di elevazione. Il segmento di controllo ha il compito di controllare e monitorare l'intero sistema GPS.

L'intera sezione è composta da una stazione di comando, la MCS (Master Control Station), situata nella Falcon Air Force Base in Colorado, da cinque stazioni di monitoraggio MS (Monitor Station) e da tre stazioni trasmittenti GA (Ground Antenna), controllate dall'organo statunitense DSCS (Defense Satellite Communications System). Le stazioni MSC e GA sono situate attorno al Globo nelle vicinanze dell'equatore. La stazione di comando è responsabile del lavoro svolto da tutto il control segment; essa elabora le informazioni pervenute da tutti i satelliti attraverso le 5 stazioni di monitoraggio, mette a punto tutte le correzioni

necessarie per ogni satellite e comanda la trasmissione delle stesse attraverso le 3 stazioni di controllo.

Tra le operazioni svolte dal Control Segment, le più importanti sono quelle di monitoraggio dello stato dei satelliti, monitoraggio degli orologi di bordo dei satelliti, del calcolo dei fattori orbitali e della trasmissione delle correzioni di tali parametri, ottenute avvalendosi di strumenti di misurazione e di calcolo notevolmente potenti e accurati, infatti la MSC è dotata di una serie di orologi atomici altamente precisi ai quali vengono riferiti tutti gli altri orologi, sia a terra che a bordo dei satelliti.

Infine il segmento utente è rappresentato da tutti gli utenti civili e militari del sistema GPS, in particolare dai ricevitori, decodificatori ed elaboratori dei segnali GPS.

La Fig. 1.3 mostra uno schema generale della struttura del sistema GPS, visto nei suoi principali segmenti.

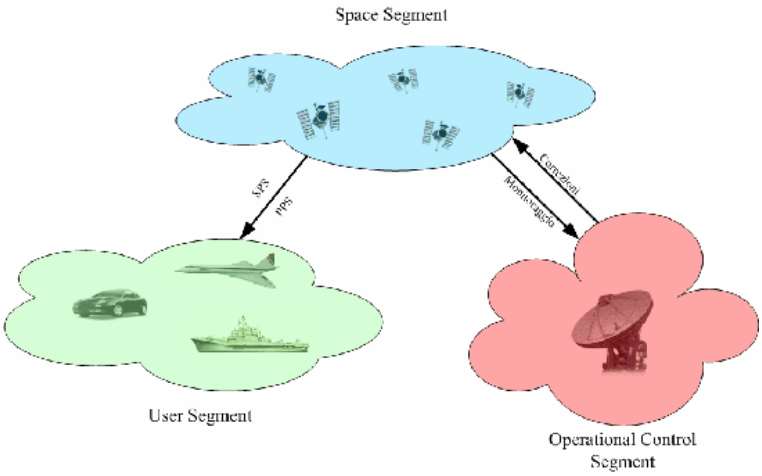


Fig. 1. 3 – Organizzazione del sistema GPS nei tre segmenti

1.3. Come funziona il GPS

Il sistema di funzionamento del GPS è relativamente semplice. Si basa, di fatto, sul metodo della triangolazione, un sistema utilizzato per secoli dai navigatori; il sistema ricevente dell'utente riceve impulsi dai satelliti della costellazione Gps e, attraverso un sistema di equazioni, desume la propria posizione triangolando i segnali automaticamente. Utilizzando la rilevazione della posizione di almeno tre

punti fissi (con coordinate note) si calcola la propria posizione, data dall'incontro delle rette passanti per detti punti. Questa prima considerazione fa dedurre, quindi, che non è il dispositivo di navigazione al suolo (in mare o nel cielo) che comunica la propria posizione ai satelliti, come si potrebbe immaginare, ma il contrario. Il dispositivo è atto a ricevere segnali univoci e continui dal satellite che li invia in maniera unidirezionale. La ricezione dei segnali di tre distinti satelliti fornisce un'indicazione abbastanza precisa della posizione, ma non assolutamente precisa; per questo motivo occorre ricevere l'impulso da un quarto satellite per ottenere la maggiore precisione possibile come vedremo tra poco. Ogni satellite della costellazione genera un segnale contenente tre informazioni: il proprio identificativo, la posizione sull'orbita in cui si trova e un segnale temporale la cui precisione è garantita dall'orologio atomico montato a bordo. Attraverso queste informazioni il dispositivo ricevente è in grado di conoscere la distanza esatta dal satellite, moltiplicando il tempo di percorrenza del segnale per la velocità della luce, circa 300.000 km al secondo. Chiaramente, però, il ricevitore può trovarsi in un qualsiasi punto di un'ipotetica sfera il cui raggio è rappresentato dalla distanza ricevitore/satellite. La ricezione di un secondo segnale, analogo al primo, da un altro satellite genererà una seconda sfera che s'intersecherà con la prima in due punti formando un'ellisse entro la quale si troverà il punto ricercato. Basterà quindi ricevere il segnale da un terzo satellite per limitare le possibilità a due punti molto vicini (uno dei quali potrà essere automaticamente eliminato per via di considerazioni matematiche e cinematiche) e quindi ottenere la corretta posizione dell'apparato ricevente. I problemi essenziali a questo punto sono due: calcolare con massima precisione il tempo di percorrenza del segnale e garantire la migliore sincronizzazione degli orologi (quello sul satellite e quello dell'apparato di ricezione). Il sistema di codici utilizzato per ottenere la migliore precisione del tempo di percorrenza è di tipo "pseudocasuale". S'intende con questo termine un sistema di codici estremamente complesso, tale da apparire pressoché casuale. In realtà si tratta di un codice ripetuto mille volte al secondo. La lettura del codice generato dal satellite e di quello generato localmente dal dispositivo ricevente (teoricamente nello stesso istante) e il loro confronto causano una discrepanza e quindi una grandezza misurabile che fornisce il valore della distanza. Il satellite emette il proprio segnale, e invia il proprio codice, su due portanti: L1 (1.575,42 MHz) e L2 (1.227,60 MHz). Sulla prima portante

viaggia un codice, detto C/A (Coarse Acquisition), ripetuto ogni 1.023 bit, e occupa una banda da 1 MHz. Il C/A definisce un'acquisizione grossolana in quanto lo sfasamento di un microsecondo (pari a poco meno di un ciclo), moltiplicato per l'enorme distanza, può generare errori fino a 300 metri. Il secondo codice P (Precise) è modulato invece a 10 MHz, quindi con una frequenza al secondo dieci volte maggiore. Questo è il codice che, combinato con il primo, garantisce la precisione richiesta. Il codice P è anche quello che l'Amministrazione americana si è riservata di potere degradare in caso di necessità e di minaccia alla sicurezza. Risolto il problema dei codici la questione è riconducibile a un problema di precisione degli orologi. I quattro montati sul satellite sono precisi al milionesimo di secondo, ma non altrettanto precisi possono essere quelli montati sull'apparecchiatura. Entra in gioco allora il segnale ricevuto dal quarto satellite. Supponendo che tutti i dispositivi (spazio-terra) siano perfettamente sincronizzati, la triangolazione di quattro segnali che s'intersecano dovrebbe fornire l'indicazione accurata di un punto. Si risolve, in pratica, un sistema di quattro equazioni in quattro incognite rappresentate dalla latitudine, longitudine, altitudine ed errore dell'orologio.

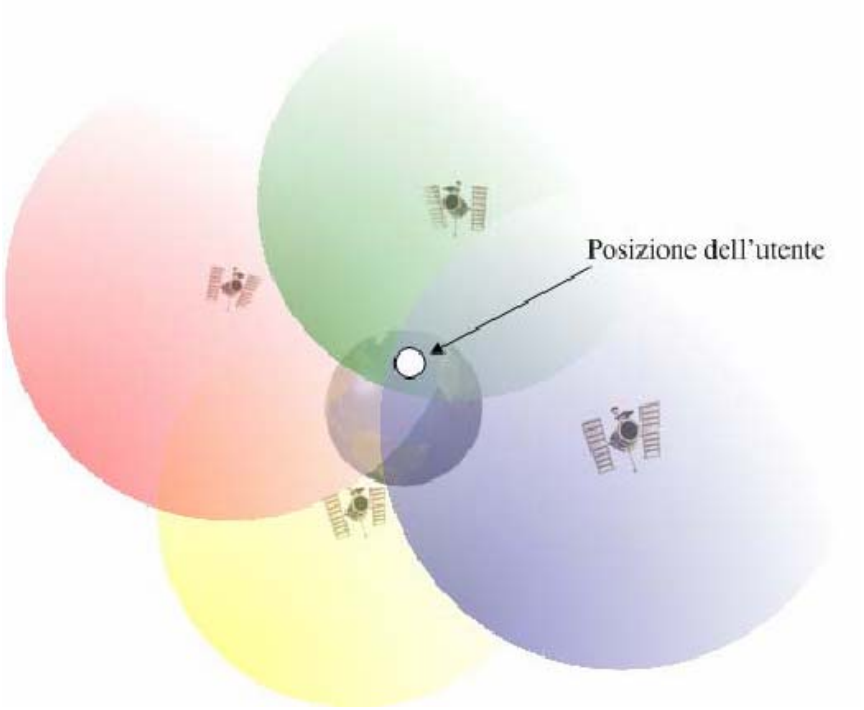


Fig. 1. 4 – Misura della posizione nelle tre coordinate spaziali con errore di sincronizzazione

1.4. Scale temporali e segnale PPS

Il GPS, come già accennato, può essere usato per la sincronizzazione temporale. Esistono diverse scale per la misurazione del tempo, e si dividono principalmente in tre categorie fondamentali: siderale, solare e atomica. Le prime due sono legate al moto di rotazione terrestre rispetto a una direzione di riferimento nello spazio: le scale siderali rispetto alla direzione dell'equinozio; le scale solari rispetto alla direzione del Sole; La direzione di riferimento può essere quella vera o una direzione media, depurata degli effetti perturbativi. Una scala di tempo siderale è la GMST (Greenwich Mean Sidereal Time): angolo orario fra meridiano di Greenwich e equinozio medio, aumentato di 12 ore; di tipo solare è la scala UT1 (Universal Time 1): angolo orario fra meridiano di Greewich e direzione del Sole, depurato degli effetti del polar motion, aumentato di 12 ore. A causa dell'irregolarità del moto di rotazione terrestre queste scale di tempo presentano irregolarità significative e quindi tempo solare e tempo siderale differiscono poiché, a causa del moto di rivoluzione terrestre intorno al Sole, la direzione di

questo vista dalla Terra retrocede di giorno in giorno (in un anno vi sono 365.2422 giorni solari e 366.2422 giorni siderali).
Le scale di tempo atomico sono invece definite da un certo numero di orologi atomici al Cesio 133 e sono le più accurate e stabili. In particolare abbiamo la scala TAI (Tempo Atomico Internazione), definita dal BIPM di Parigi; la scala UTC (Tempo Universale Coordinato) che ha la stessa cadenza del TAI ma è periodicamente aggiornata sottraendogli 1 s per mantenerla sincronizzata entro il secondo con la scala UT1. Infine abbiamo la scala di tempo GPST (GPS Time), che è quella a cui il sistema, e quindi il ricevitore, GPS fa riferimento. Quest'ultima coincide con il TAI a meno di un offset, definito in modo che al 6 gennaio 1980, ore 00.00, il GPST coincidesse con l'UTC.
Attraverso il ricevitore GPS si può quindi accedere con la precisione di una scala di tempo atomica, ricevendo un particolare segnale ogni secondo chiamato PPS, che può essere utilizzato come sistema di riferimento per il tempo. Il segnale PPS garantisce una precisione dell'ordine di pochi nanosecondi ed è in assoluto il migliore standard disponibile per lunghi tempi di integrazione.

1.5. Errori del sistema

Fino a questo punto si è ipotizzato che la posizione derivata dal sistema GPS sia molto precisa ma in realtà esistono varie fonti d'errore che diminuiscono la precisione della posizione GPS di alcuni metri. Molti di essi possono essere corretti mediante appositi algoritmi o formule derivate da misure effettuate durante la fase sperimentale del sistema.
Le fonti principali di errore sono:

- **Ritardi d'origine ionosferica ed atmosferica :** Il segnale GPS attraversando l'atmosfera può subire un rallentamento, con un effetto simile alla luce che si rifrange attraverso un blocco di vetro. Tale rallentamento può introdurre un elemento di errore nel calcolo della posizione a terra in quanto la velocità del segnale viene alterata. La ionosfera non causa un ritardo costante del segnale, ma esistono vari fattori che contribuiscono ad influenzarne l'effetto, come l'elevazione del

satellite, l'influenza del Sole sulla densità della ionosfera e il vapore acqueo contenuto nell'atmosfera. Questi effetti però sono controllabili attraverso l'uso di ricevitori GPS a doppia frequenza.

- **Errori degli orologi di satellite e ricevitore :** benché gli orologi operativi sul satellite siano molto accurati (fino a circa 3 nanosecondi), sono soggetti in alcuni casi a lievi variazioni che danno luogo ad errori di minima entità che influiscono quindi sull'accuratezza della posizione. Il Dipartimento della Difesa statunitense controlla gli orologi dei satelliti mediante il segmento di controllo e può correggere qualsiasi variazione rilevata.

- **Multipath :** Il Multipath del segnale GPS si verifica quando l'antenna del ricevitore è posizionata vicino ad un'ampia superficie riflettente, quale per esempio un lago o un edificio. Il segnale del satellite non si dirige direttamente sulla antenna, bensì colpisce l'oggetto vicino e viene riflesso sull'antenna, dando luogo quindi ad una falsa misurazione. Gli effetti del Multipath possono essere ridotti utilizzando delle speciali antenne GPS che incorporano un "ground plane", un disco metallico, che impedisce all'antenna la ricezione di segnali provenienti dal basso. Le antenne GPS dell'ultima generazione sono in grado di filtrare il segnale che ha subito un Multipath anche se la massima accuratezza è ottenibile utilizzando un'antenna di tipo choke ring. Questo tipo di antenna è infatti dotata di 4 o 6 anelli concentrici che "intrappolano" i segnali indiretti. Il Multipath influisce solamente sulle misure di precisione, del tipo richiesto dai rilievi topografici. I semplici ricevitori portatili per navigazione non utilizzano tecniche di filtro per i segnali riflessi.

- **Diluizione della precisione :** la Diluizione della Precisione (DOP) è il parametro di valutazione della disposizione dei satelliti e si riferisce alla distanza e alla posizione dei satelliti nel cielo. Il parametro DOP può amplificare l'effetto degli errori di rilevamento dei satelliti. Quando i satelliti sono correttamente distanziati la posizione è individuabile nell'ambito dell'area in ombra del diagramma e il margine di errore

possibile è ridotto. Quando i satelliti sono eccessivamente ravvicinati le dimensioni dell'area in ombra aumentano e aumenta di conseguenza anche l'incertezza della posizione. I diversi tipi di Diluizione della Precisione sono calcolabili in base alle dimensioni. Il miglior modo per ridurre al minimo gli effetti del DOP consiste nell'osservare il maggior numero possibile di satelliti.

- **Disponibilità selettiva e Anti Spoofing (S/A) :** La disponibilità selettiva insieme all' anti spoofing è un algoritmo, inizialmente applicato al segnale GPS dal Dipartimento della Difesa statunitense, allo scopo di impedire ai civili e a potenze straniere ostili di usufruire al completo della capacità di localizzazione del sistema GPS; il processo consiste nell'introdurre una piccola alterazione nel funzionamento degli orologi dei satelliti. Inoltre le effemeridi (ovvero la posizione del satellite sul percorso) vengono trasmesse in modo leggermente diverso dal reale. Ne risulta la degradazione dell'accuratezza della posizione. Recentemente, come già detto in precedenza, questi algoritmi sono stati eliminati, consentendo ai ricevitori civili un incremento della precisione.

1.6. Il ricevitore GPS

I ricevitori GPS sono ricevitori radio sintonizzati sulle frequenze usate dal sistema, dotati di sistemi di decodifica ed elaborazione dei segnali ricevuti e di una memoria per l'immagazzinamento dei dati. Sono composti da un'antenna che viene disposta sul punto da determinare. L'antenna è collegata mediante un cavo schermato al ricevitore propriamente detto, ovvero al gruppo di sintonizzazione e acquisizione dati che comprende oltre ai sistemi di decodifica del segnale e al microprocessore di elaborazione, un orologio di precisione (normalmente un oscillatore al quarzo di elevata qualità), una memoria fisica interna e un software interno per il controllo del processo di acquisizione dati. Le caratteristiche dei ricevitori variano a seconda della ditta produttrice e del modello. Le differenze più significative fra i vari tipi di ricevitori attualmente in commercio sono le seguenti:

- possibilità di ricezione dei segnali su una sola frequenza, (apparati "monofrequenza") o entrambe le frequenze disponibili;
- numero di canali di ricezione (ne occorre uno per satellite), da un minimo di 4 a un massimo di 12;
- misura con il metodo "pseudoranges" o anche con il metodo per "differenza di fase".

Il ricevitore utilizzato in questo lavoro è il *Trimble Resolution T* (Fig. 1.5) che per alcune proprietà elencate di seguito è stato scelto tra i vari modelli presenti sul mercato.

Il *Resolution T* ha un numero di canali in ricezione pari a 12 ed opera sulla frequenza L1 (1575.42 MHz). Sfrutta inoltre lo standard SPS usando il codice C/A (Coarse Acquisition). Necessita di una doppia alimentazione: 3.3V per alimentare il GPS e 5.5V per alimentare l'antenna. Il ricevitore è montato su di una motherboard che permette l'interfacciamento con un PC attraverso la porta seriale RS-232, ed è stato scelto in primo luogo per l'alta risoluzione nelle applicazioni temporali infatti, prevede una precisione dell'ordine di pochi nanosecondi (1 Sigma) per l'uscita PPS.

Un'altra importante caratteristica è data dall'antenna (Fig. 1.6) che ha forma semisferica per permettere di correggere l'errore di multipath già descritto nel paragrafo precedente, evitando quindi di incorrere in errori dovuti alla ricezione di segnali riflessi. Può inoltre operare usando uno dei due protocolli disponibili: TSIP (Trimble Standard Interface Protocol) o NMEA 0183 (National Marine Electronics Association). Il primo, utilizzato in questo lavoro, è un protocollo a pacchetti binari che permette all'utilizzatore il massimo controllo sulla configurazione, fornendo come dato in uscita anche l'errore di quantizzazione che, permette di ottenere una correzione sul segnale PPS emesso dal ricevitore e quindi una più alta precisione. Il protocollo NMEA invece è uno standard industriale usato soprattutto per applicazioni marine e permette una compatibilità diretta con altri dispositivi che supportano questo standard.

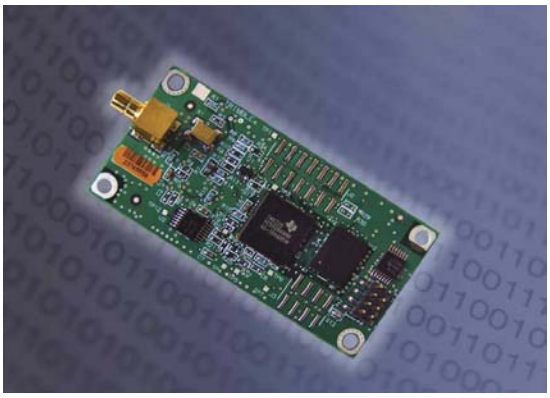


Fig. 1.5 – Ricevitore resolution T

Fig. 1. 6 – Antenna del ricevitore

Capitolo 2

Esperimento EEE e rivelatori

2.1. L'esperimento EEE

L'esperimento EEE (Extreme Energy Events) ha come obiettivo lo studio della radiazione cosmica ad alta energia. Esso prevede che nel corso dei successivi tre anni scolastici, attraverso una sinergia tra Scuola, Università ed Enti di ricerca, 21 Scuole del territorio nazionale (Licei e Istituti Tecnici) vengano dotate di un apparato sperimentale dedicato all'osservazione e alla misura dei muoni cosmici, consistente in un "telescopio" formato da tre piani di rivelatori del tipo Multigap Resistive Plate Chambers (MRPC).
Il progetto si articola nelle seguenti tre fasi:

1. costruzione dei rivelatori MRPC;
2. realizzazione del telescopio con MRPC e messa a punto della strumentazione;
3. presa dati e analisi.

Questo lavoro si propone di sviluppare una parte della terza fase del progetto, più precisamente si occupa della rivelazione dei dati catturati dai telescopi affrontando tutte le problematiche esistenti in un sistema con dati dislocati.

2.2. Il telescopio a MRPC

Il sistema di rivelazione modulare del Progetto EEE, che viene installato in ogni Scuola, è un telescopio costituito da tre piani di rivelatori MRPC (Fig. 2.1). Ogni

camera, che offre un'area sensibile di (1.6 x 0.82) m^2, presenta una struttura a sandwich, costituita da:

- due piani esterni di elettrodi metallici che hanno la forma di strisce longitudinali, ciascuna lunga 1.6 m e larga 34 mm ;
- una coppia di vetri resistivi, cui è applicata una alta tensione, posta tra i due piani;
- un insieme di lastre di vetro, posizionate all'interno, ed intervallate da strati di gas, nel quale i raggi secondari rilasciano la propria energia ionizzandone le molecole.

Il sistema descritto è capace di misurare con grande precisione il punto d'impatto della particella cosmica incidente e il suo tempo di attraversamento. La precisione nella determinazione della coordinata trasversale del punto d'impatto è di 34 mm, ma potrà anche risultare migliore nel caso in cui due strip vicine diano segnale. Ogni strip è connessa, a ciascuna delle sue estremità, con un sistema elettronico di lettura e di acquisizione del segnale. La differenza in tempo tra i segnali ai due estremi di ogni strip produce la coordinata longitudinale del punto d'impatto, con una precisione di circa 1 cm. Tramite la misura della posizione dei tre punti d'impatto (uno per piano) è quindi possibile ricostruire la traiettoria rettilinea della particella che ha attraversato il telescopio. Per la lettura e l'acquisizione dei dati, a ogni telescopio è associata una catena elettronica costituita da: un sistema di front end, per l'amplificazione e la discriminazione dei segnali forniti dagli elettrodi di readout dei rivelatori MRPC; da un sistema di conversione, per la digitalizzazione delle informazioni acquisite e da un sistema di trigger, per la selezione delle particelle, che genera un segnale (trigger), quando almeno una striscia di ogni singolo piano è attraversata da una particella. Il segnale di trigger si ottiene effettuando un OR logico fra tutti i segnali provenienti dalle strisce di ogni singolo piano e, successivamente, ponendo i tre OR in AND logico. La catena elettronica è connessa con un calcolatore tramite un'opportuna interfaccia. Il telescopio è dunque in grado di acquisire dati e di trasmetterli via rete ad un opportuno "centro di raccolta". Ogni telescopio inoltre è geograficamente localizzato e temporalmente sincronizzato via satellite tramite un sistema GPS. È dunque prevista anche l'installazione di un'apposita antenna GPS. Così facendo i telescopi delle varie Scuole possono essere messi in coincidenza in fase di analisi dei dati, allo scopo di rivelare eventi cosmici

di energie estreme connessi a sciami cosmici di grande apertura angolare; in questo modo il notevole numero di muoni da essi trasportati, provenienti da un punto comune nell'alta atmosfera terrestre (il cosiddetto vertice d'interazione del raggio cosmico primario che ha dato origine allo sciame) potrebbero essere rivelati simultaneamente da diversi telescopi situati a grande distanza l'uno dall'altro. I dati trasmessi da tutti i telescopi nelle varie Scuole saranno raccolti e archiviati presso il CNAF (National Center for Telematics and Informatics) dell'INFN (Istituto Nazionale di Fisica Nucleare) di Bologna. L'analisi dei dati sarà effettuata tramite il sistema innovativo di calcolo distribuito GRID, usufruendo dell'esperienza del CERN e dell'INFN in tale settore.

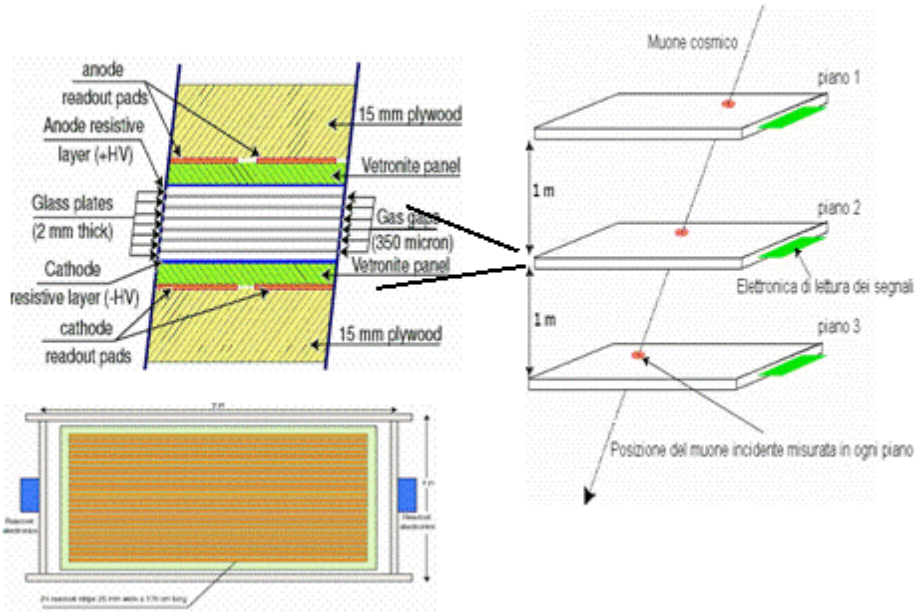


Fig. 2.1 – Principio di funzionamento di un rivelatore MRPC

2.3. Teoria dell'operazione

La fase del progetto da espletare in questa tesi, come già detto precedentemente, riguarda la rilevazione dei dati e una prima analisi, consistente nell'organizzazione temporale dei dati rilevati sull'intero territorio nazionale.
I dispositivi fin qui esposti sono caratterizzati dal fatto di essere esterni al modulo progettato, più precisamente sono dei dispositivi di input del sistema. Di seguito è riportato uno schema a blocchi che descrive le varie parti del progetto e i collegamenti tra questi:

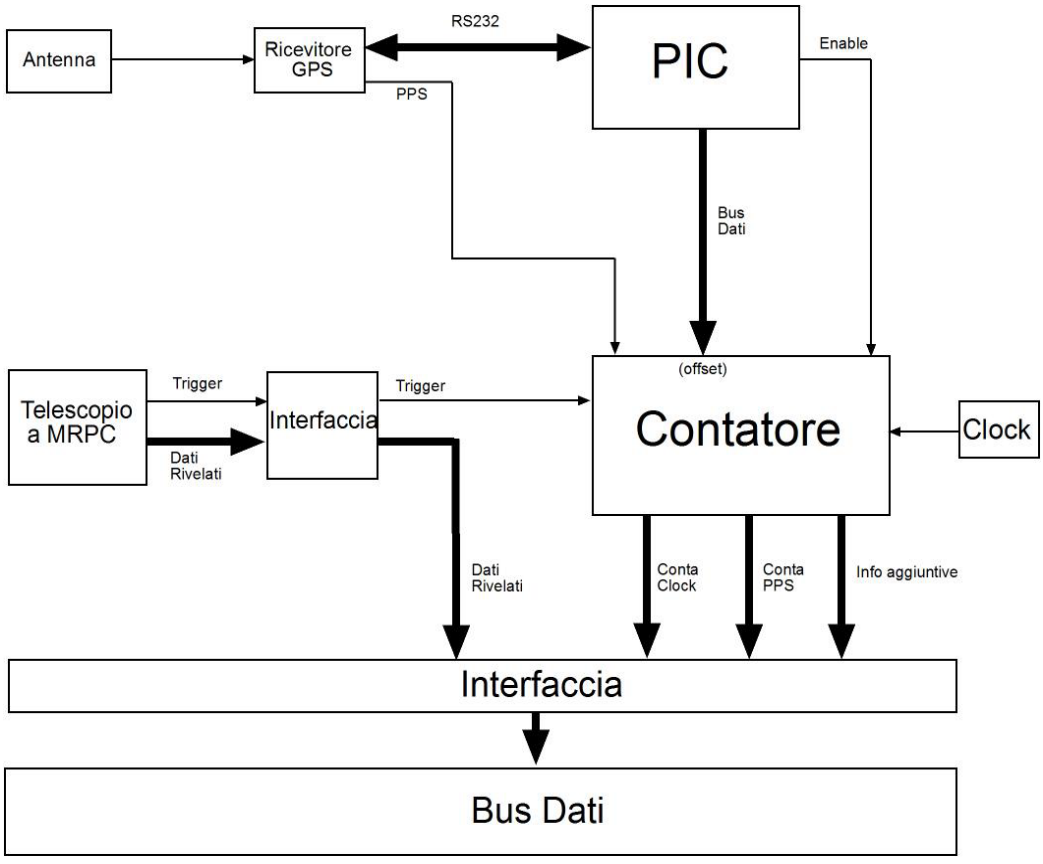


Fig. 2.2 – Schema a blocchi del dispositivo generale

I dispositivi esterni sono: antenna, ricevitore GPS, telescopio a MRPC.
Gli altri blocchi invece fanno parte di un unico circuito che estrae i dati acquisiti e li organizza dal punto di vista temporale, creando un database elettronico.
Essi sono:

- il blocco PIC, che corrisponde ad un microcontrollore della *Microchip,* che estrae i dati necessari dal ricevitore GPS, per ottenere un riferimento temporale con una precisione del secondo, converte l'UTC e la data in secondi dall'inizio dell'anno e, associando questo valore a posizione del GPS ed errore di quantizzazione, invia il tutto tramite un bus dati, al dispositivo successivo;
- il blocco CONTATORE, che in realtà contiene una coppia di contatori. Questo dispositivo, si può dire, che permette di colmare la lacuna della PIC, permettendo una precisione superiore al secondo, attraverso l'uso di un clock esterno, indicato in figura, con frequenza pari a 50 Mhz.

Il ricevitore GPS invia i dati alla PIC attraverso la porta seriale RS232 senza necessità di un'interfaccia, visto che il ricevitore è montato su una scheda madre che prevede la conversione dei dati dallo standard TTL a quello RS232. I dati inviati alla PIC sono in formato TSIP, ed oltre a questi viene anche fornito il segnale PPS. La PIC ha il compito di estrarre solo i dati utili, per questo è stato necessario interpretare il protocollo TSIP, in modo da elaborare solo le stringhe di codice che sono di nostro interesse. Il dispositivo CLOCK, che è visto in figura come un blocco esterno al contatore, è costituito in pratica da un oscillatore al quarzo. Nel circuito di test è stato utilizzato un oscillatore al quarzo stabilizzato in temperatura, affiancato da un oscillatore non stabilizzato, perché il primo consente di ottenere un'alta precisione, e quindi una maggiore stabilità dei dati da analizzare. Praticamente il quarzo stabilizzato è visto come un clock "quasi" ideale, usato solo nella fase sperimentale, nella realizzazione finale verrà utilizzato il quarzo non stabilizzato, dato l'alto costo del concorrente associato al gran numero di dispositivi da realizzare. L'oscillatore stabilizzato, incorpora al suo interno, un dispositivo che permette, attraverso un sensore di temperatura, di raggiungere una determinata temperatura e mantenerla costante nel tempo, per garantire la stabilità del clock. Si intuisce che questo quarzo ha bisogno di un certo tempo per raggiungere la temperatura prefissata e stabilizzarsi. Il quarzo non stabilizzato invece, subisce delle variazioni sui cicli di clock, di un range

prestabilito ed indicato dalla casa produttrice, dovute all'aumento di temperatura. Il blocco contatore, permette quindi di aumentare la precisione del time stamping dei dati, riempiendo la "gap" che lascia la PIC, corrispondente ad un range di un secondo, attraverso il clock fornito dal quarzo stabilizzato. Il contatore riceve dalla PIC, il numero di secondi dall'inizio dell'anno, quando questo dato è disponibile, ponendo il segnale "enable" nello stato alto e i segnali PPS, che fungono da riferimento per il conteggio. Uno dei contatori fa riferimento al valore dei secondi dall'inizio dell'anno, come punto di partenza, per poi incrementarlo grazie agli impulsi PPS. Il secondo contatore, conta le oscillazioni del clock fornito dal quarzo, azzerando il valore ogni secondo. Il segnale di trigger stabilisce il momento in cui questi dati (comprensivi di errore di quantizzazione e posizione del GPS) devono essere riportati in uscita, insieme ai dati rivelati dal telescopio, ottenendo quindi un riferimento temporale di questi ultimi, con una precisione dell'ordine dei nanosecondi.

Il contatore fornisce in uscita i dati relativi a posizione ed errore di quantizzazione, tramite la linea dati indicata in figura 2.2, come "info aggiuntive". Per concludere quindi, la PIC ci fornisce un riferimento temporale universale, scandendo il tempo con intervalli del secondo, mentre il contatore permette, tramite una sincronizzazione con la PIC, una precisione dell'ordine dei nanosecondi, necessaria al time stamping dei dati.

Sono stati utilizzati un contatore ed una PIC con le varie interfacce perché questa soluzione garantisce la massima integrazione rapportata alla massima velocità di esecuzione dei vari processi.

Infine, tutti i dati necessari per una successiva elaborazione ed organizzazione sono inviati ad un'interfaccia che è collegata ad un bus dati. Data l'alta frequenza di lavoro del circuito (50 Mhz) e la dimensione delle stringhe da elaborare (fino a 26 bit per stringa), sarà necessario usare all'interno dell'interfaccia finale un buffer per la memorizzazione temporanea dei dati, per non perdere nessun ciclo.

Nei prossimi due capitoli si analizzeranno in dettaglio i due blocchi fondamentali del dispositivo per capirne meglio il funzionamento, elencando le caratteristiche dei singoli componenti utilizzati, i problemi affrontati nella realizzazione e le soluzioni possibili.

Capitolo 3

Sviluppo del firmware della PIC

3.1. Il protocollo TSIP

Il ricevitore GPS trasmette dei pacchetti di dati con tutte le informazioni relative alla posizione, al tempo e alle richieste effettuate.

Il *Resolution T* , come già accennato, può fornire questi dati usando uno dei protocolli disponibili tra l' NMEA ed il TSIP. Il protocollo utilizzato in questo esperimento è il TSIP che rispetto al primo fornisce, su richiesta, il valore dell'errore di quantizzazione per ogni segnale PPS ricevuto. Sarà chiaro nel prossimo paragrafo l'importanza della presenza dell'errore di quantizzazione tra i dati prelevati dal GPS.

Il TSIP inoltre, rispetto all' NMEA, permette la configurazione della porta seriale RS232 in modo da avere un controllo bidirezionale, attraverso l'invio di richieste al GPS. Si può così avere un controllo sui dati ricevuti e soprattutto si può evitare di ricevere dati inutili che ritardano solo la lettura di quelli utili.

Il protocollo TSIP è basato quindi, sulla trasmissione di pacchetti di informazioni tra il dispositivo collegato e il ricevitore GPS. Ogni pacchetto include un codice che identifica il significato e il formato dei dati trasmessi nel suo interno, ed è preceduto e seguito da alcuni caratteri di controllo. Il GPS è configurato per trasmettere automaticamente in uscita i pacchetti 0x8F-AB e 0x8F-AC (0x indica che i valori sono indicati in esadecimale). Per la maggior parte degli utilizzi del GPS questi pacchetti sono sufficienti perché contengono i dati relativi al tempo, alla posizione e allo stato e funzionamento del GPS.

I parametri di configurazione del GPS sono memorizzati in una memoria non volatile. L'utente può cambiare il valore dei parametri per ottenere l'operazione desiderata, ma le modifiche dei parametri non vengono memorizzate

automaticamente, per far questo deve inviare il pacchetto 0x8E-26 direttamente al *Resolution T,* che provvederà a salvare le modifiche in memoria. Per ripristinare i parametri al valore di default, basta inviare il pacchetto 0x1E al ricevitore.
La velocità con cui i dati vengono trasmessi con questo protocollo è di 9600 baud, sia in uscita che in entrata, inoltre questi ultimi hanno una lunghezza di 8 bit; la parità è dispari, con un bit di stop.
La struttura dei pacchetti TSIP è la stessa sia per le richieste che per i dati in uscita, ed è di seguito indicata:

<DLE> <id> <data string bytes> <DLE> <EXT>

Dove:

<DLE> rappresenta il byte 0x10
<EXT> rappresenta il byte 0x03
<id> rappresenta il byte che identifica il pacchetto, il quale può avere qualsiasi valore tranne <EXT> e <DLE>.
La "stringa dati" può avere qualsiasi valore. Considerando valida l'ultima affermazione, possiamo supporre che all'interno della "stringa dati", sia presente il valore "<DLE> <EXT>", questo però può portare a confondere una sequenza di valori all'interno della stringa dati, con la sequenza che indica la fine del pacchetto. Per ovviare a questo inconveniente, il protocollo TSIP utilizza un algoritmo apposito:
ogni byte <DLE> contenuto nella stringa dati è preceduto da un byte <DLE> extra, che deve essere aggiunto prima dell'invio del pacchetto e rimosso dopo la sua ricezione. Grazie a questo algoritmo si può identificare la fine del pacchetto, che è rappresentata dal byte <EXT> preceduto da un numero dispari di byte <DLE>.
Di seguito è riportato un semplice esempio, per comprendere meglio il funzionamento dell'algoritmo:

<DLE> <id> <DLE> <EXT> <DLE> <EXT> stringa dati senza algoritmo

<DLE> <id> <DLE> <DLE> <EXT> <DLE> <EXT> stringa dati con algoritmo

La prima sequenza ricevuta è sempre quella di inizio pacchetto, corrispondente a <DLE> <id>, successivamente si ha la stringa dati (evidenziata in rosso) che, nel primo caso non prevede l'algoritmo di correzione, ed essendo identica alla sequenza di fine pacchetto, non c'è modo di identificarla univocamente. Nel secondo caso invece, è applicato l'algoritmo e, anche se all'interno della stringa dati è presente la sequenza <DLE> <EXT> identica a quella di fine pacchetto, non può mai essere confusa con quest'ultima, visto che all'interno della stringa dati il valore <EXT> è preceduto da un numero pari di <DLE>.
I pacchetti che contengono tutte le informazioni necessarie da fornire al sistema progettato sono i due che il GPS trasmette per default, cioè 0x8F-AB e 0x8F-AC, ed un terzo, ottenuto tramite la richiesta 0x24 che restituisce il pacchetto 0x6D. Si analizza ora ogni singolo pacchetto per individuare i dati necessari tra tutti quelli presenti, in modo da poterne effettuare l'estrazione.
Di seguito vengono riportate tre tabelle con tutti i dati contenuti in ogni singolo pacchetto:

Byte	Bit	Item	Type	Value	Description
0		Subcode	UINT8		0xAB
1 - 4		Time of week	UINT32		GPS seconds of week
5 - 6		Week number	UINT16		GPS week number
7 - 8		UTC Offset	SINT16		UTC Offset (seconds)
9	0	Timing Flag	Bit field	0	GPS time
				1	UTC time
	1			0	GPS PPS
				1	UTC PPS
	2			0	Time is set
				1	Time is not set
	3			0	Have UTC info
				1	No UTC info
	4			0	Time from GPS
				1	Time from user
10		Seconds	UINT8	0 - 59	Seconds
11		Minutes	UINT8	0 - 59	Minutes
12		Hours	UINT8	0 - 23	Hours
13		Day of Month	UINT8	1 - 31	Day of month
14		Month	UINT8	1 -12	Month of year
15 - 16		Year	UINT16		Four digits of year

Tabella 3.1: pacchetto ricevuto 0x8F-AB

Byte	Item	Type	Value	Description
0	Subcode	UINT8	0xAC	
1	Receiver mode	UINT8	0 - 7	
2	Reserved	UINT8	0 - 6	Reserved
3	Self-survey progress	UINT8		0 - 100%
4 - 7	Reserved	UINT32	0	Reserved
8 - 9	Reserved	UINT16	0	Reserved
10 - 11	Minor alarms	UINT16		
12	GPS decoding status	UINT8		
13	Reserved	UINT8	0	Reserved
14	Spare status 1	UINT8	0	
15	Spare status 2	UINT8	0	
16 - 19	Local clock bias			ns
20 - 23	Local clock bias rate			ppb
24 - 27	Reserved			Reserved
28 - 31	Reserved			Reserved
32 - 35	Temperature			Degrees C
36 - 43	Latitude	Double		Radians
44 - 51	Longitude	Double		Radians
52 - 59	Altitude	Double		Meters
60 - 63	PPS Quantization Error			Seconds
64 - 67	Spare			Future expansion

Tabella 3.2: pacchetto ricevuto 0x8F-AC

Byte	Bit	Item	Type	Value	Meaning
0	0 : 2	Fix dimension	Bit field	1 - 5	
0	3	fix mode	Bit field	0 - 1	Auto-manual
0	4 : 7	Number of sv's in fix	Bit field	0 - 12	Count
1 - 4		PDOP	Single		PDOP
5 - 8		HDOP	Single		HDOP
9-12		VDOP	Single		VDOP
13-16		TDOP	Single		TDOP
17- n		SV PRN	SINT8	+/- (1-32)	PRN

Tabella 3.3: pacchetto ricevuto 0x6D

Ogni pacchetto è composto da un numero variabile di byte nei quali sono contenute diverse informazioni diversificate da un item, nella terza colonna è indicato il tipo di informazione che si sta trasmettendo, e nella successiva i valori che questa può assumere. Nell'ultima colonna si ha una breve descrizione utile alla comprensione dei dati. Il byte "0" è l'identificativo del pacchetto, infatti nella colonna "value" ritroviamo il nome di quest'ultimo.

Da 0xF-AB, che rappresenta il pacchetto primario del tempo, si sono estratti tutti i dati per ottenere i secondi dall'inizio dell'anno e l'anno corrente, più precisamente, nella tabella 3.1, dal byte "10" al "14" corrispondenti rispettivamente a: secondi, minuti, ore, giorno del mese e mese. Nei byte "15 – 16" è contenuto l'anno corrente.

Dal pacchetto 0x8F-AC si può ricavare la posizione del GPS attraverso i dati relativi alla latitudine (byte "36 – 43"), longitudine ("44 – 51") e altitudine ("52 – 59"). In questo pacchetto inoltre, è contenuta l'informazione relativa all'errore di quantizzazione indicata nella tabella 3.2 in corrispondenza dei byte "60 – 63". Nel pacchetto 0x6D, inviato dal GPS su richiesta, l'unica informazione necessaria da estrarre riguarda il numero di satelliti connessi, che come si è già visto precedentemente, deve essere superiore o uguale a 4 per ottenere dei dati attendibili.

Attraverso la prima porta seriale del GPS, questi dati sono inviati alla PIC per una successiva elaborazione.

Direttamente dal ricevitore GPS invece, più esattamente dal pin 9 della seconda porta seriale, otteniamo infine l'impulso positivo PPS, che è capace di pilotare un carico fino a 5mA senza danneggiare il ricevitore.

3.2. L' errore di quantizzazione

Il flusso di dati in qualsiasi processore è riferito alla velocità di questo ed è misurato in MHz. Un singolo ciclo di questa frequenza è di solito riferita ad un ciclo di clock, ed ogni istruzione o calcolo avviene in corrispondenza di un ciclo. Così per ottenere il segnale PPS ci deve essere un ciclo di clock. Questo metodo di riferimento dei segnali in corrispondenza del clock introduce un errore chiamato errore di quantizzazione.

Il ricevitore GPS *resolution T* garantisce un'alta affidabilità del segnale PPS, grazie all'accuratezza degli orologi atomici a cui fa riferimento, ma ciò nonostante anche questo segnale, come tutti i segnali reali, è affetto da diversi errori. Quando le applicazioni richiedono maggiore precisione di quella fornita dal ricevitore GPS a basso costo, uno dei principali parametri su cui agire è quello relativo alla correzione dell'errore di quantizzazione.

Il *resolution T* include un oscillatore con frequenza a 12.504 MHz, quindi il ciclo di clock si ha 12.504.000 volte al secondo, che corrisponde ad un periodo di 80 nanosecondi tra due cicli di clock. Possiamo pensare quindi che il GPS per 80 nanosecondi è incapace di fornire il PPS. Il ricevitore tipicamente pone il segnale PPS nel ciclo di clock vicino; ad esempio, se si suppone che GPS idealmente, fornisce il PPS 50 nanosecondi dopo il ciclo di clock, allora deve aspettare e portare in uscita il PPS al prossimo ciclo di clock. Questo ritardo in realtà non è mai costante, quindi gli 80 nanosecondi di gap introducono un errore di $\pm$ 40 nanosecondi (Fig. 3.1). La forma d'onda rappresenta il clock a 12.504 MHz. L'uscita del *resolution T* deve essere posta su uno dei fronti del clock, quindi l'ultima forma d'onda è quella che si ha in uscita dal GPS, mentre la seconda rappresenta il segnale del tempo UTC ricevuto. La differenza in termini di tempo tra il fronte di salita del clock usato per generare il PPS e il punto in cui il *resolution T* ha determinato dove dovrebbe essere posto il PPS, rappresenta la misura dell'errore

di quantizzazione. Usando entrambi i fronti del clock l'errore sul PPS per il *resolution T* scende a ± 20 nanosecondi.

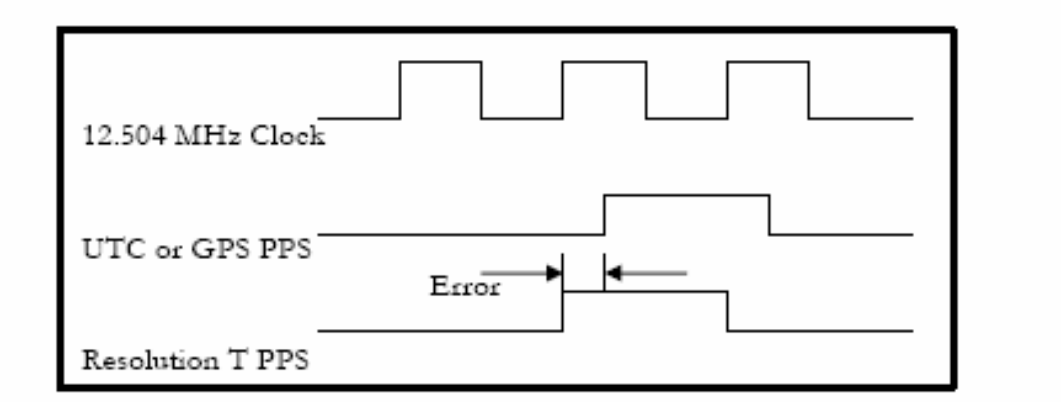


Fig. 3.1 – Errore di quantizzazione

La misura dell'errore di quantizzazione può essere usata quindi, per ridurre l'effettivo jitter sull'impulso PPS.

Le specifiche del *resolution T* riportano una precisione sul tempo del valore di 1-sigma, che corrisponde ad un valore inferiore ai 15 nanosecondi. Rimuovendo l'errore di quantizzazione dal segnale PPS la precisione aumenta, raggiungendo un valore inferiore ai 5 nanosecondi.

Sono presenti molti altri errori che influenzano la precisione, ma questi sono quasi irrilevanti rispetto all'errore di quantizzazione.

Un procedimento per eliminare l'errore di quantizzazione ma non implementato in questo caso, consiste nell'utilizzo di un PLL (phase locked loop), cioè di un anello ad aggancio di fase, che mette in sintonia la frequenza dell'oscillatore con quella del sistema GPS, permettendo una riduzione dell'errore sulla fase del segnale. Nel diagramma (Fig. 3.2) è riportato lo schema a blocchi di un circuito PLL: al nodo "A" si ha il segnale PPS che include l'errore di quantizzazione, al nodo "B" l'errore di quantizzazione riportato dal GPS attraverso la porta seriale. Il nodo "C" corrisponde all'errore di fase dopo aver sottratto l'errore di quantizzazione. In figura (Fig. 3.2) è riportato il grafico temporale del segnale ai tre nodi descritti; si vede come l'errore in fase è notevolmente ridotto.

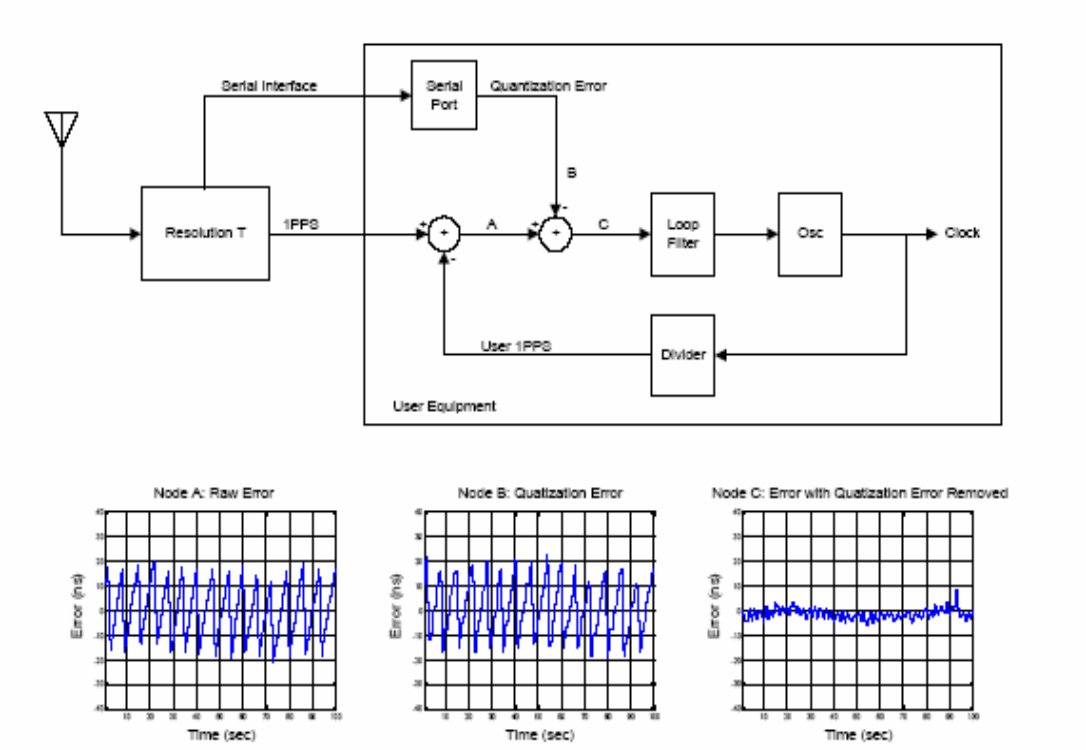


Fig. 3.2 – Schema a blocchi di un PLL e grafici dei segnali ai nodi A, B e C.

3.3. Programmazione della PIC

Le informazioni trasmesse dal GPS, come già detto, sono organizzate in pacchetti, tramite il protocollo TSIP. L'estrazione dei dati da questo protocollo è il compito assegnato alla PIC, che collegata tramite una porta seriale al ricevitore GPS, ed opportunamente programmata, estrae i dati necessari da fornire ai successivi dispositivi. Il modello di PIC utilizzato in questo lavoro è la PIC16F877 (appendice A).

In un primo momento, si è collegato il ricevitore GPS in ingresso alla PIC, mentre l'uscita di quest'ultima era collegata ad un display LCD in modo da poter verificare, l'esattezza dei dati estratti e soprattutto la presenza di errori o ritardi nella lettura. Tramite il software messo a disposizione dalla *Trimble*, è stato possibile visualizzare sul PC, i pacchetti ricevuti dal GPS. Questo ha permesso lo studio in tempo reale di ogni singolo pacchetto e la successiva implementazione del codice per programmare la PIC.

I dati contenuti nei pacchetti vengono visualizzati e aggiornati dal software sul PC, con cadenza di un secondo. Di seguito è riportato un esempio di acquisizione dati:

Receive Packet ID: 8F-AB Data Length: 17 (da GPS a PC)
AB 00 04 6E 89 05 5D 00 00 08 29 28 08 03 05 07 D6

Receive Packet ID: 8F-AC Data Length: 68 (da GPS a PC)
AC 07 00 00 00 00 00 00 00 08 00 00 00 00 00 C8 8E DA AD 43 6D 65 13 00 00 00 00 00 00 00 00 41 E2 FA D0 3F E6 87 19 2C 02 49 2A 3F D4 3A F2 31 AF 7B 75 40 50 84 23 55 74 00 00 32 11 E8 FE 00 00 00 00

Transmit Packet ID: 24 Data Length: 0 (da PC a GPS)

Transmit Packet ID: 27 Data Length: 0 (da PC a GPS)

Receive Packet ID: 6D Data Length: 20 (da GPS a PC)
38 00 00 00 00 00 00 00 00 00 00 00 3F 80 00 00 13 16 0E

Receive Packet ID: 47 Data Length: 61 (da GPS a PC)
0C 01 BF B3 33 33 15 80 00 00 00 10 80 00 00 00 03 80 00 00 00 1C 80 00 00 00 13 40 D9 99 9A 16 40 86 66 66 08 80 00 00 00 0E 40 86 66 66 12 80 00 00 00 1A 80 00 00 00 1D 80 00 00 00

Come si può notare, i pacchetti corrispondono alla descrizione fornita dal protocollo utilizzato. I primi due sono i pacchetti ricevuti per default dal GPS, poi si ha una richiesta da parte del PC di due pacchetti supplementari, corrispondente al comando “transmit packet”, che vengono forniti subito dopo. Ad esempio se vogliamo ricavare l’orario in cui questi dati sono stati ricevuti, basta convertire l’undicesimo valore del primo pacchetto da esadecimale in decimale per ottenere i secondi, ed i due successivi rispettivamente, per minuti ed ore. Nella ricezione dei pacchetti, si è osservato che la loro lunghezza non è costante, come ci si potrebbe aspettare dal fatto che le informazioni al loro interno sono divise in byte corrispondenti, nelle tabelle dei pacchetti, univocamente a quel dato. Questa variazione proviene dal fatto

che, come osservato in precedenza, ogni volta che il GPS riceve all'interno della stringa dati, il valore 0x10, che corrisponde anche al valore di inizio e fine del pacchetto, questo viene seguito da un valore identico extra, in modo da non confondere il valore interno alla stringa che rappresenta un dato, con quello utilizzato dal protocollo per la separazione dei pacchetti. Per questo, se all'interno di una stringa dati sono presenti due valori 0x10, il pacchetto 0x8F-AB che normalmente ha una lunghezza pari a 17 byte assumerà una lunghezza pari a 18 byte. Tutti i byte nella stringa dati, successivi al dato 0x10, quindi vengono sfasati di una posizione rispetto a quella che normalmente occupano, e così via se si presenta un altro dato 0x10. Questo naturalmente ha portato ad affrontare la stesura del codice per la PIC tenendo conto del fatto che l'array che immagazzina i dati può essere variabile e soprattutto che i dati non sono sempre nella stessa posizione. Nell'appendice B è riportato il listato del codice di programmazione della PIC.

Si esaminano di seguito le istruzioni eseguite dalla PIC, riportate, per chiarire ulteriormente i passaggi, in un diagramma di flusso (Fig. 3.3) :

prima di tutto vengono inizializzate le linee di I/O della PIC, definendo per ogni porta se si tratta di ingresso o uscita e se la PIC gestisce segnali analogici o digitali.

Successivamente, si definiscono le variabili e si verifica l'esatto funzionamento del display LCD, collegato in uscita alla PIC, attraverso la visualizzazione di una stringa: "dati provenienti dal GPS". Ora bisogna estrarre le informazioni necessarie dai pacchetti ricevuti dal GPS: vengono inizializzate alcune variabili; attraverso un comando di "wait" si attende che la PIC riceva dal GPS l'ID del pacchetto utile, corrispondente alla prima informazione del pacchetto. Individuata l'intestazione del pacchetto, seguono tutte le informazioni così come vengono riportate nelle tabelle precedentemente descritte. Per ovviare al problema del dato "0x10", si estrae un numero di byte superiore a quello previsto dal pacchetto; si potrebbe estrarre un numero di byte in più molto alto, nell'eventualità che il dato "0x10" si ripeta molte volte nella stringa dati, ma facendo questo rischiamo, nel caso in cui ad esempio il dato "0x10" non sia presente, di acquisire in un array i dati del pacchetto successivo, che dovrebbero essere contenuti in un altro array. Per questo si prevede un numero di byte in più non superiore a tre, per non "intaccare" il pacchetto successivo. Questo vuol dire che, quando i dati del pacchetto contengono un numero di byte "0x10" superiore a tre, si può verificare un errore nell'acquisizione dati. Il problema può essere risolto semplicemente, inserendo un algoritmo di controllo che verifichi la

successione dei dati e che corregga l'errore quando i valori non sono consecutivi. Il primo pacchetto estratto è quello che il ricevitore fornisce in uscita per primo, e cioè il pacchetto 0x8F-AB. Questo contiene i dati relativi a secondi, minuti, ore, giorno, mese ed anno, che vengono inseriti in un array. Per conoscere la posizione esatta dei singoli dati nell'array, dobbiamo eliminare l'eventuale presenza dei byte "0x10" superflui. Questo viene fatto attraverso una concatenazione di cicli IF che controllano la presenza di due valori "0x10" consecutivi, eliminandone uno, ed inserendo tutto il contenuto dell'array in un secondo array "normalizzato" che non contiene cioè i dati superflui. Il limite pari a tre per i byte in più da acquisire, è dovuto al fatto che il primo pacchetto contiene l'informazione sull'anno corrente in ultima posizione quindi, dopo di questo abbiamo la sequenza <DLE> <EXT> <DLE>, dove i primi due termini indicano la fine del pacchetto in esame, mentre l'ultimo indica l'inizio di quello successivo. Per questo l'acquisizione di un byte in più potrebbe non far funzionare il "wait" sul pacchetto successivo.

Le operazioni eseguite sul pacchetto seguente (10x8F-AC) sono identiche a quelle precedenti e permettono l'estrazione dei dati relativi a: latitudine, longitudine, altitudine ed errore di quantizzazione. Queste informazioni vengono sempre inserite in un array successivamente "normalizzato".

L'ultimo pacchetto estratto (0x6D) ottenuto in risposta dal GPS alla richiesta effettuata dalla PIC, contiene le informazioni relative al numero di satelliti collegati. Questo dato, come gia anticipato, è necessario per verificare l'attendibilità dei dati fin ora acquisiti. Infatti i dati possono essere considerati esatti solo se il numero di satelliti collegati è compreso fra 4 e 12. Il numero minimo di satelliti collegati è posto pari a quattro poiché per misurare in modo esatto e contemporaneamente latitudine, longitudine, altitudine e tempo, sono necessarie quattro variabili e quindi quattro segnali ricevuti in questo caso da diversi satelliti.

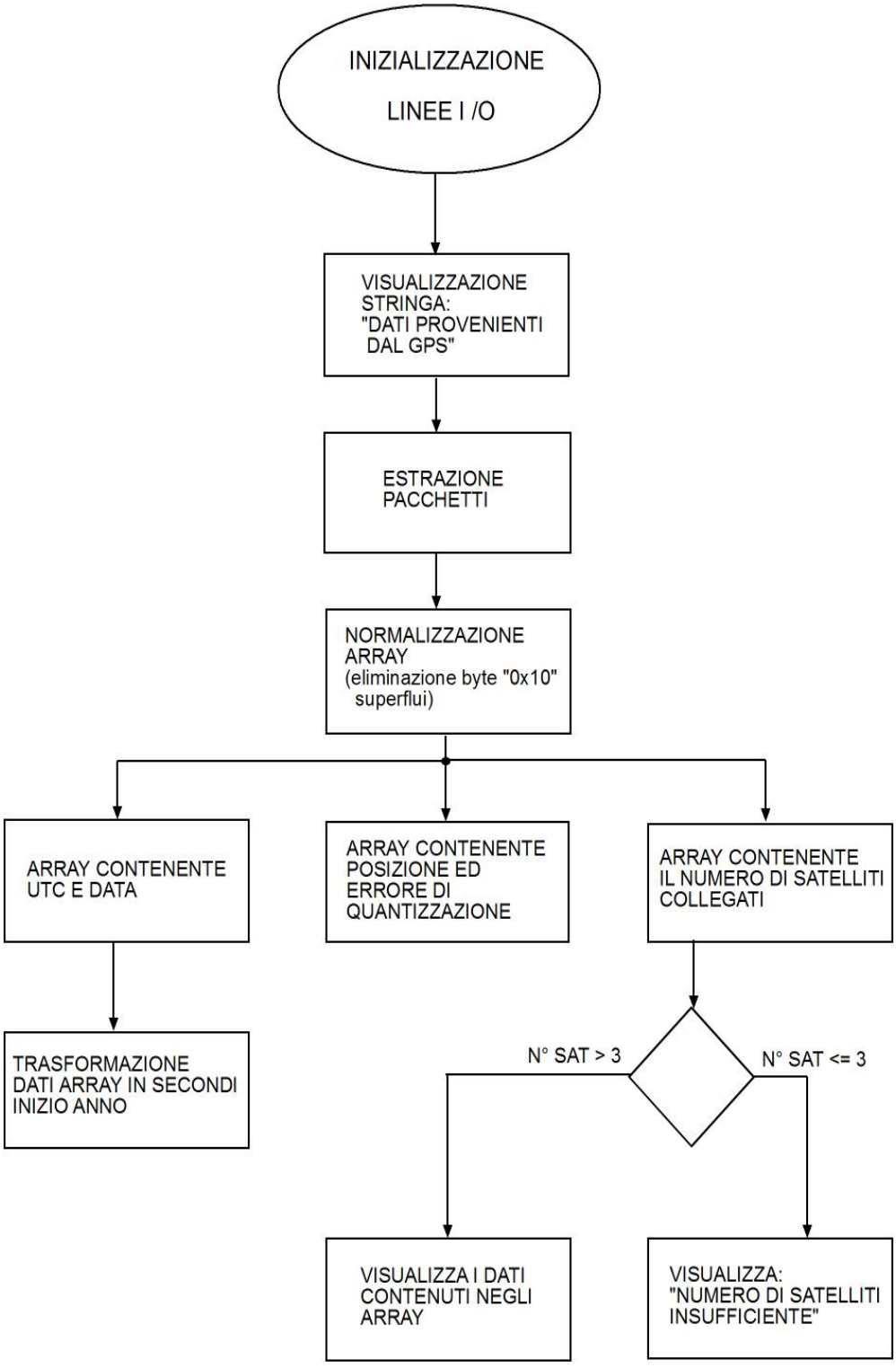


Fig. 3.3 – Diagramma di flusso delle operazioni eseguite dalla PIC

Dalle tabelle dei singoli pacchetti si possono ricavare i tipi di dati usati nel protocollo TSIP:

- secondi, minuti, ore, giorno del mese e mese sono byte di tipo UINT8. Questo vuol dire che il valore rappresentato è composto da 8 bit ed è privo di segno, può assumere i valori compresi tra 0 e 255;
- latitudine, longitudine ed altitudine, sono indicati come tipi "Double", cioè possono assumere i valori compresi tra $1.7x10^{-308}$ e $3.4x10^{+308}$, con 53 bit di precisione;
- l'errore di quantizzazione è un dato di tipo "Single" e può assumere i valori compresi tra $3.4x10^{-38}$ e $1.7x10^{+38}$, con 24 bit di precisione;
- il numero di satelliti collegati è un dato di tipo "bit field" e usa quattro bit per rappresentare un numero compreso tra 0 e 12.

L'ultima operazione eseguita dalla PIC riguarda la trasformazione dell' orario UTC ricevuto in secondi dall'inizio dell'anno. Per far questo, prima di tutto si verifica attraverso un ciclo il caso in cui si ha un anno bisestile, in modo da poter considerare il mese di febbraio di 28 o 29 giorni. Successivamente si ricavano i giorni trascorsi dell'anno corrente (variabile "dayanno") che vengono poi trasformati in ore (variabile "temp"). Il passaggio da ore a secondi dall'inizio dell'anno richiede l'uso di due variabili, perché essendo quet'ultimo un numero relativamente grande per essere contenuto in una variabile "word", bisogna usarne due. Il numero di secondi contenuti in un anno, è pari a 31536000, che corrisponde ad un numero binario di 25bit. Poiché una variabile "word" può contenere al massimo 16 bit, se ne usano due, facendo in modo che, attraverso un operazione di assegnazione, si abbiano i bit più significativi in una variabile "word" chiamata "secannoh", mentre i bit meno significativi vengono memorizzati in una variabile "word" chiamata "secannol". Bisogna ora aggiungere a questo valore diviso in due variabili i minuti trasformati in secondi ed i secondi trascorsi, così si avrà il conteggio dei secondi dall'inizio dell'anno. I minuti convertiti in secondi sommati ai secondi trascorsi, sono memorizzati in una variabile chiamata "secanno". Si deve ora decidere come fare per sommare questa quantità ad un valore che è diviso in due variabili. Una variabile "word" può rappresentare un valore decimale massimo pari a 65535; il numero massimo di secondi contenuto in un ora, che corrisponde al massimo valore che può

assumere la variabile “secanno”, vista la sua definizione, è pari a 3600. Per ottenere la somma quindi, basta controllare se la quantità “secanno” può essere contenuta nella variabile “secannol” e se la verifica ha esito positivo, basta solo fare la somma tra le due, altrimenti oltre a questo bisogna incrementare di uno la variabile che contiene i bit più significativi e cioè “secannoh”.
Per concludere, viene effettuata una verifica dei dati attraverso la visualizzazione di questi su di un display LCD.

3.4. Test del codice per la PIC

Una volta programmata, la PIC viene inserita in un circuito creato per il test del codice ottenuto (Fig. 3.4). La programmazione avviene attraverso la porta seriale del PC collegata al circuito di test, tramite un deviatore che in base alla posizione riceve o trasmette i dati dalla porta. Il ricevitore GPS può essere inoltre collegato tramite la porta seriale sia al circuito di test che al PC. Il collegamento al circuito permette di verificare il funzionamento del programma, attraverso la visualizzazione dei dati a display. Il collegamento al PC permette di verificare che la PIC trasmetti al ricevitore GPS la richiesta del pacchetto aggiuntivo necessario all’estrazione del numero di satelliti collegati e soprattutto di confrontare i valori visualizzati sul display LCD con quelli ottenuti dal software della *Trimble* per verificare l’esattezza dei dati ottenuti. I valori visualizzati sul display, quando è garantito il minimo numero di satelliti collegati, sono:

- UTC
- Data
- Latitudine
- Longitudine
- Altitudine
- Numero di satelliti collegati
- Errore di quantizzazione

Se le informazioni non risultano attendibili, sul display è visualizzata la stringa: “Numero di satelliti insufficiente”.

Quando il numero di satelliti collegati è superiore a tre, il circuito estrae i dati in modo continuo e senza errori. Il collegamento ad un numero di satelliti superiore a tre, è garantito dal ricevitore dopo un tempo di qualche minuto dal momento dell'accensione, ed è mantenuto se le condizioni di ricezione sono ottime. Il circuito prevede 8 bus di uscita da 16 pin ciascuno, per trasferire le informazioni prelevate, al blocco successivo.

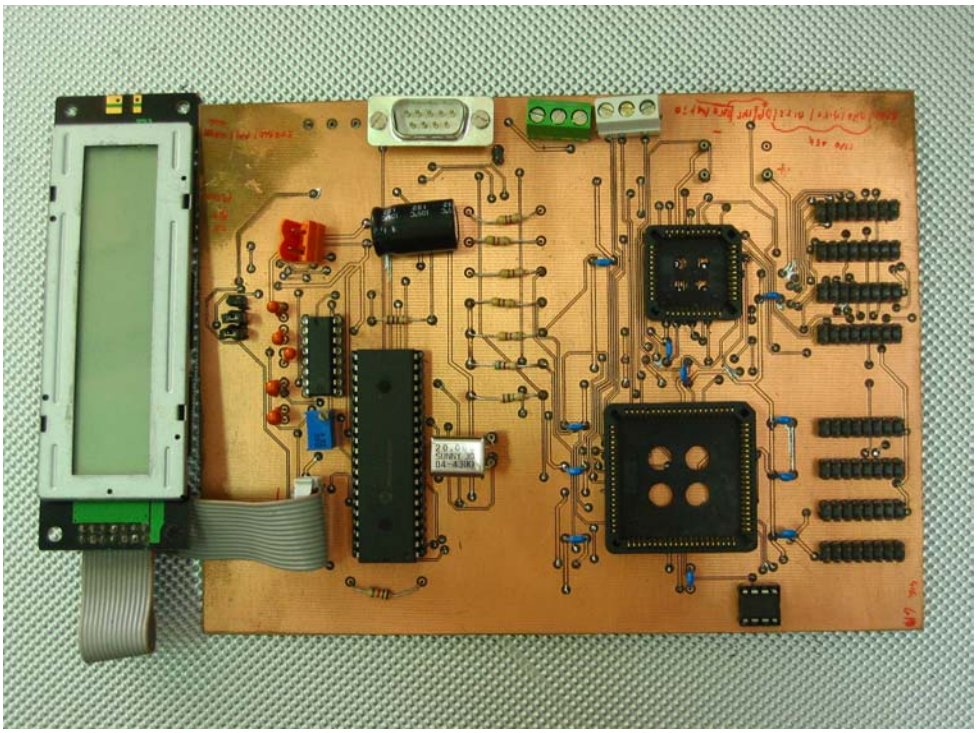


Fig. 3.4 – Circuito di test per la PIC

Capitolo 4

Sviluppo del firmware dell'FPGA

4.1. Introduzione ai dispositivi FPGA e teoria dell'operazione

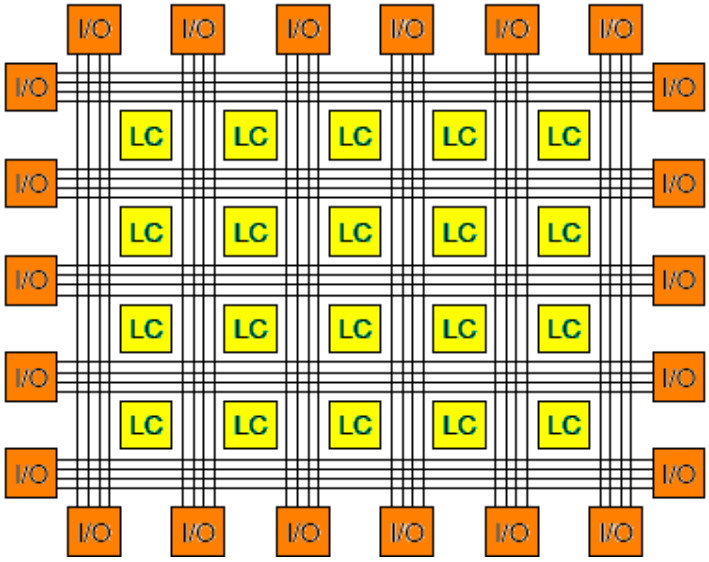


Fig. 4.1 – Diagramma a blocchi di una FPGA generica

Le FPGA (Field Programmable Gate Array) sono dispositivi programmabili dall'utente, costituiti da un array di componenti logici, circondati da blocchi di I/O programmabili e collegabili tra loro tramite interconnessioni anch'esse programmabili; le FPGA costituiscono un'importante evoluzione nel mondo dei dispositivi programmabili poichè forniscono un'elevata potenza di calcolo e di

connessione. Un aspetto molto importante sono i collegamenti locali che attraversano il dispositivo e che sono condivisi da pochi
elementi logici, per cui la potenza utilizzata ed i ritardi che si generano sono limitati. All'interno di uno stesso dispositivo possono esserci differenti linee locali di lunghezza diversa e questo garantisce un'elevata flessibilità del sistema.
Field Programmable Gate Array, significa che la funzione dell' FPGA è definita dal programma dell'utente, piuttosto che dalla disposizione, non modificabile, dei dispositivi che realizzano le funzioni logiche.
Questi dispositivi permettono di raggiungere livelli di integrazione molto spinti, mantenendo la caratteristica di basso costo di produzione iniziale, tipico dei dispositivi programmabili.
Esistono diverse FPGA in commercio con caratteristiche molto variabili. In questo lavoro si è utilizzata una SPARTAN-XL della XILINX, perché soddisfa le richieste di progetto, corrispondenti a:

- Un adeguato numero di connessioni I/O, vista la dimensione ed il numero dei dati in gioco;
- Un'alta velocità di esecuzione delle istruzioni da eseguire;
- Capacità di porte equivalenti disponibile;
- Possibilità di programmazione sul campo.

Il modello di SPARTAN-XL utilizzata è, più precisamente, la XCS10XL che ha le seguenti caratteristiche:

Dispositivo	**Celle logiche**	**Massimo numero di Gate**	**Range tipico Di Gate**	**Matrice CLB**	**CLB totali**	**Numero FLIP FLOP**	**I/O**	**Bit di RAM totali**
XCS10XL	466	10.000	3.000 – 10.000	14 x 14	196	616	112	6.272

Le altre caratteristiche (data sheet), come piedinatura e altro, sono riporatate nell'appendice C.

Il codice viene trasferito nell'FPGA saldata sulla scheda da una EEPROM montata su zoccolo e opportunamente programmata.

Come si vede dalle caratteristiche di questa FPGA, il numero di piedini disponibili come ingressi o uscite è pari a 112, un numero abbastanza elevato per questo tipo di applicazione. Le celle logiche disponibili per la programmazione sono 466, che corrisponde ad una quantità molto più alta di quella necessaria alla realizzazione del blocco da progettare ma utile per eventuali sviluppi futuri. Altre caratteristiche sono il numero di CLB (Configurable Logic Blocks), pari a 196, corrispondenti ai quadratini in giallo indicati con la sigla LC di figura 4.1; la matrice è inoltre composta da 14 righe e altrettante colonne. Il numero di flip-flop implementabili al suo interno è pari a 616.

La rete implementata nella FPGA deve assolvere il compito di doppio contatore, come già detto precedentemente. Di seguito è riportato uno schema a blocchi, che visualizza i segnali di ingresso, di uscita e le varie interconnessioni tra i due contatori implementati nella FPGA:

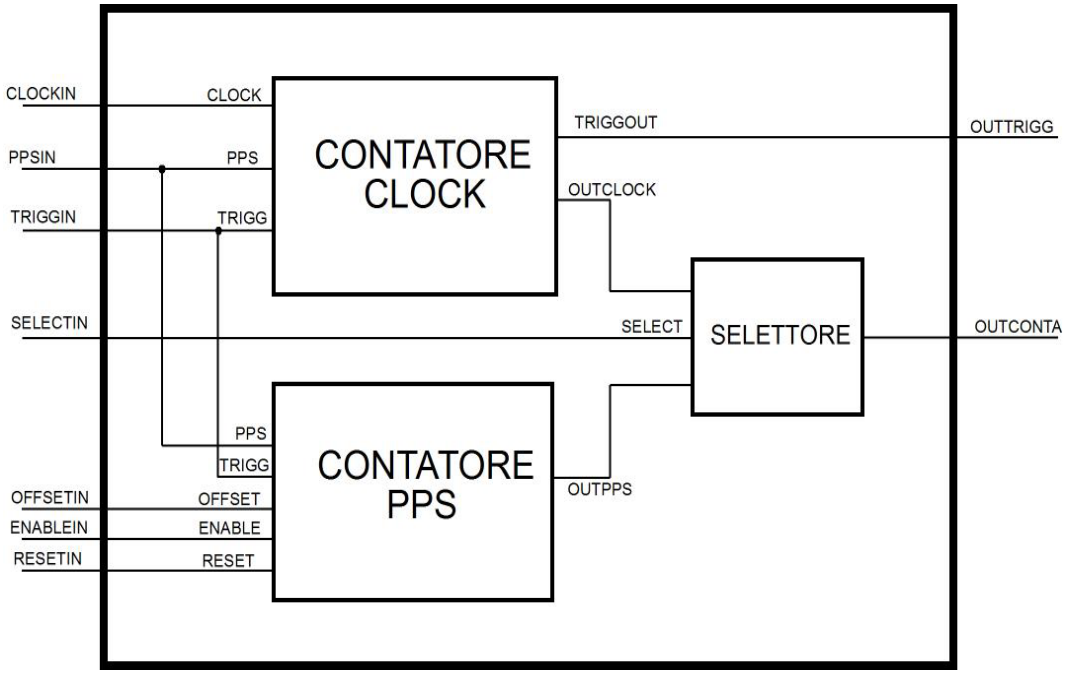


Fig. 4.2 – Schema a blocchi del contatore realizzato con FPGA

Questo schema è una visione più dettagliata del blocco "CONTATORE" indicato in figura 2.1. Vediamo come questo viene integrato nello schema generale:

- il segnale "clockin", corrisponde all'ingresso proveniente dal blocco "clock";
- "ppsin" proviene direttamente dal ricevitore GPS e corrisponde al segnale PPS prelevato dalla porta seriale;
- "triggin" arriva in ingresso al contatore, dall'interfaccia collegata al telescopio a MRPC, corrispondente al segnale di trigger, cioè di dato pronto in ingresso al contatore;
- il segnale "offsetin", corrisponde al bus di dati proveniente dalla PIC;
- entrambi i segnali "enablein" e "resetin", vengono impostati dall'utilizzatore, per effettuare determinate operazioni di lettura o di reset;
- "selectin" è semplicemente un segnale che permette di selezionare uno dei due segnali provenienti dai contatori, per portarlo in uscita;
- l'uscita "outconta" corrisponde ai due segnali in uscita al contatore dello schema generale "contaclock" e "contapps".

Si vedrà in seguito che uno stesso segnale può effettuare operazioni logiche diverse a seconda del contatore che pilotano e che quindi non si può etichettate una variabile con la sua funzionalità.

4.2. Cenni sui linguaggi utilizzati per FPGA

Esistono diversi linguaggi per la programmazione di dispositivi FPGA, ma il software che la casa produttrice *Xilinx* fornisce per la programmazione delle SPARTAN-XL utilizza i due linguaggi più diffusi. Più che due linguaggi, in realtà si tratta di un unico codice utilizzato con approcci diversi:

- **Schematic**
- **VHDL**

Il primo è un metodo di programmazione grafico che permette un'ampia visibilità dei dispositivi e delle porte utilizzate, e quindi una maggiore facilità di comprensione del funzionamento. Il VHDL invece è di per se un linguaggio di programmazione vero e proprio, composto da una sequenza di codice scritto. Anche se molto più complesso del primo il VHDL permette l'integrazione

massima, dovuta all'assenza di codice ridondante. Quindi il metodo migliore per programmare una FPGA è attraverso il VHDL che permette di personalizzare il codice a livelli molto alti, diminuire le celle utilizzate ed aumentare quindi la velocità di esecuzione. La particolarità del VHDL sta nel fatto che diversamente da molti altri linguaggi, l'interpretazione del codice non è sequenziale ma concorrenziale. Questo vuol dire che tutte le istruzioni vengono eseguite parallelamente. In sostanza il VHDL è un linguaggio descrittivo, che descrive appunto un circuito elettronico attraverso la connessione di porte logiche e l'utilizzo di diversi segnali. Esso rappresenta uno standard industriale per la descrizione, la modellizzazione e sintesi di circuiti e sistemi digitali.
Gli elementi fondamentali della sintassi del codice sono:

- **entità** (ENTITY);
- **architettura** (ARCHITECTURE)

L'entità corrisponde ad un blocco funzionale, ed è equivalente ad un simbolo in uno schema elettrico. Nella figura 4.2 i blocchi "CONTATORE CLOCK", "CONTATORE PPS", "SELETTORE", insieme al blocco generale che li contiene, sono delle entità. In essa si definiscono soltanto le linee di input ed output ed il tipo di dato ricevuto o trasmesso per ciascun segnale. Ad ogni entità è associata un'architettura che descrive il comportamento di questa, attraverso l'uso di funzioni, processi e variabili interne.
Questa premessa sul linguaggio VHDL è necessaria per la comprensione del codice sviluppato per la SPARTAN-XL.

4.3. Sviluppo del codice per FPGA

Una grossa pecca individuata nell'utilizzo dell'FPGA della *Xilinx* è stata quella di non poter utilizzare l'ultima versione del software disponibile (ISE Foundation 8.1i), perché non supporta il modello di FPGA utilizzato. La versione utilizzata, che lo implementa, è "Xilinx Foundation F1.5".
L'operazione affidata a questo blocco circuitale, come già detto precedentemente, è quella di un doppio contatore, necessaria per ottenere la precisione temporale

dell'ordine dei nanosecondi, grazie soprattutto anche al clock di riferimento ottenuto dal quarzo a 50 MHz. In un primo momento i due contatori sono stati sviluppati separatamente, per avere una visibilità più chiara dei segnali in fase di sperimentazione; successivamente i due contatori sono stati inclusi in un unico blocco funzionale.

Il primo contatore implementato nel codice (appendice D) è chiamato "CONTACLOCK", nome proveniente da quello della sua entità. Come si può vedere anche dalla figura 4.2, questo blocco è costituito da quattro segnali di ingresso e due di uscita:

- pps (ingresso);
- clock (ingresso);
- trigg (ingresso);
- reset (ingresso);
- triggout (uscita);
- outclock (uscita).

Il funzionamento logico è il seguente:

Il segnale "clock" proveniente dal blocco esterno CLOCK, cioè dall'oscillatore al quarzo (fig. 4.2), incrementa un contatore di un passo per volta, ad ogni fronte di salita del segnale. Il risultato di questo conteggio è trasferito in uscita "outclock", quando è attivo il segnale "trigg", cioè quando il valore del segnale è posto alto. Questo segnale proviene dall'interfaccia collegata con il telescopio e assume il valore alto quando i dati rivelati da quest'ultimo sono presenti in uscita. Il segnale "pps" arriva dal GPS e corrisponde ad un impulso della durata di qualche nanosecondo che si verifica ogni secondo. Ad ogni impulso ricevuto dal GPS il contatore viene azzerato. Il segnale di uscita "triggout" non è altro che il segnale "trigg", normalizzato con un clock cioè, essendo il "trigg" un segnale che può essere alto per un tempo variabile, si procede facendo in modo che questa durata sia fissa e corrispondente alla durata di un ciclo del segnale "clock". Questa operazione è effettuata da un'entità chiamata "debounce", opportunamente richiamata e inizializzata. Le conseguenze della normalizzazione del segnale "trigg", sono visualizzate in figura 4.3.

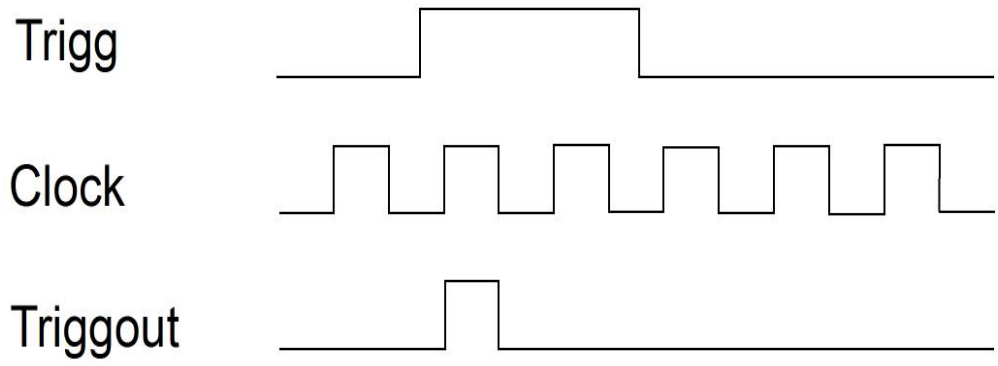


Fig. 4.3 – Effetto della normalizzazione sul segnale "trigg".

Infine il segnale "reset", quando è al livello logico alto, azzera tutti i valori dei segnali interni e delle uscite, per riportare allo stato iniziale il processo in caso di un blocco accidentale. Il contatore è implementato nell'architettura del "contaclock", attraverso una funzione di incremento di un numero binario ed una serie di processi contenenti dei cicli.

Il secondo blocco da analizzare, è un contatore chiamato "CONTAPPS". La sua entità è composta dai seguenti segnali di ingresso / uscita:

- pps (ingresso):
- trigg (ingresso);
- offset (ingresso);
- enable (ingresso);
- reset (ingresso);
- OUTpps (uscita).

Il funzionamento logico del blocco è il seguente:

in questo caso il segnale che incrementa il contatore non è più il clock del quarzo a 50 MHz ma i segnali "pps" provenienti dal ricevitore GPS. Quindi, ogni volta che si verifica un fronte di salita sul segnale "pps" un contatore interno viene incrementato di un passo. Il segnale "trigg", proveniente dall'interfaccia con il telescopio a MRPC, assolve la stessa funzione svolta nel precedente contatore cioè trasferisce in uscita, attraverso il segnale "OUTpps", il risultato dell'avanzamento del contatore. Il contatore però ha un offset iniziale non nullo e

dipendente dal segnale "offset" in ingresso, che rappresenta i secondi dall'inizio dell'anno provenienti dall'uscita della PIC. Quindi il conteggio ha inizio dal valore del segnale "offset" e avanza di un passo ad ogni fronte di salita del "pps". Il segnale "enable" permette di azzerare il contatore in modo da far partire il conteggio dall'eventuale offset. La necessità di usare il segnale "enable" nasce dal fatto che, prima che la PIC invii l'informazione relativa ai secondi dall'inizio dell'anno, questa ha bisogno di un certo tempo per agganciare il numero di satelliti necessari e ottenere quindi dei dati attendibili; una volta che questo è avvenuto, la PIC attiva il segnale "enable", azzera il contatore e fornisce la base da cui questo comincia ad avanzare.

L'ultimo segnale in ingresso è il "reset", che corrisponde allo stesso segnale del contatore precedente, ed ha anche la stessa funzione, cioè quella di azzerare tutti i segnali in caso di necessità o per un blocco accidentale del sistema.

Questa operazione di conteggio è stata implementata nell'architettura dell'entità "contapps", attraverso l'uso di due funzioni, una di incremento e l'altra di somma di due numeri binari, di una serie di processi e di cicli IF. La funzione incremento è stata utilizzata per ottenere l'avanzamento continuo di un passo ad ogni fronte di clock, mentre la funzione somma permette di sommare il valore del contatore all'eventuale valore di offset. In questo modo il segnale "enable" al livello logico alto azzera il contatore che continua però il suo avanzamento, e quando un segnale di trigger è inviato in ingresso, si riporta in uscita la somma dei valori di contatore ed offset.

Il blocco "SELETTORE" non è una vera e propria entità, ma è semplicemente un'operazione implementata nell'entita "GENERALE", nella quale avvengono tutti i collegamenti necessari tra segnali di ingresso, contatori e segnali di uscita. L'operazione di selezione in uscita del valore di uno o l'altro contatore è stata introdotta per una questione di facilità di analisi del dispositivo, e soprattutto per il fatto che i pin fisicamente collegati alla FPGA e disponibili come uscite nel circuito di test, non sono in numero sufficiente. Il selettore permette di ottenere in uscita il valore di uno dei due contatori; più precisamente se il segnale "select" è attivo basso, in uscita si ha il valore del contatore "contaclock", mentre se è attivo alto, il valore in uscita corrisponde al contatore "contapps".

4.4. Simulazione al PC del codice sviluppato

Una volta scritto il codice per ognuno dei contatori, il passo successivo consiste nel verificare l'esattezza sia logica che funzionale dei blocchi. Il software utilizzato permette di effettuare la simulazione del circuito, senza dover programmare necessariamente la SPARTAN-XL. Questo permette di poter verificare passo passo il codice scritto e assicurarsi che la funzionalità voluta è realmente implementata. La simulazione permette di applicare dei segnali in ingresso ai blocchi logici e di verificare il relativo andamento delle uscite. Grazie alla simulazione è stato possibile visualizzare una serie di ritardi nel circuito che in qualche modo ne potevano alterare il funzionamento. Si osserva infatti nella simulazione del blocco precedentemente accennato, chiamato "debounce", che effettua la normalizzazione dei segnali, un ritardo del segnale di uscita rispetto a quello di ingresso, in funzione del clock. Infatti idealmente, i segnali dovrebbero avere un andamento come quello indicato in figura 4.3. In realtà la simulazione visualizza un ritardo del segnale di uscita di un fronte di salita del clock (Fig. 4.4):

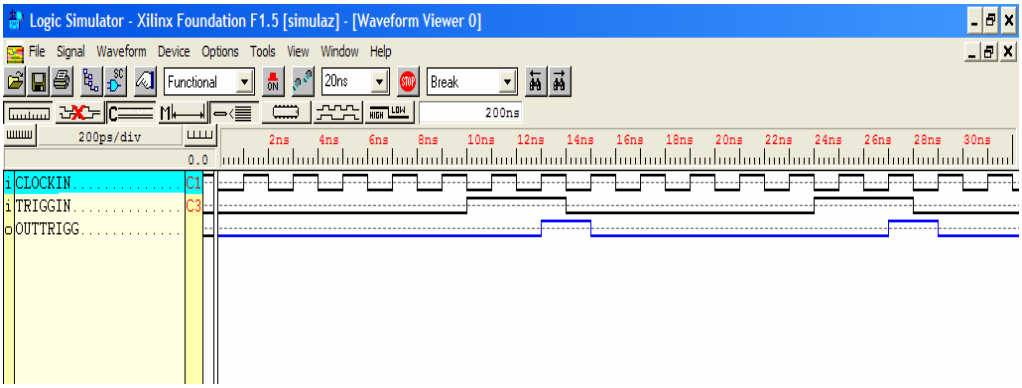


Fig. 4.4 – ritardo applicato ad un segnale dalla funzione "debounce" (il primo segnale rappresenta il clock, il secondo l'ingresso, l'ultimo è il segnale di uscita).

Il fattore di ritardo osservato non influisce sul funzionamento perché è un errore sistematico e quindi facilmente eliminabile nella fase finale. Inoltre, per alcuni segnali, il "debounce" è stato applicato solo in fase sperimentale per ovviare ad inconvenienti, mentre in fase di realizzazione pratica questo non sarà necessario.

Ad esempio il segnale "enable" è forzato ad un livello alto in fase di test, per un ciclo di clock; questo naturalmente si può avere solo tramite la funzione "debounce".

In figura 4.5 è mostrata la simulazione del blocco "contaclock"; il segnale "pps1" rappresenta il segnale "pps" dopo il debounce, ed è questo che deve essere considerato nell'analisi logica del circuito.

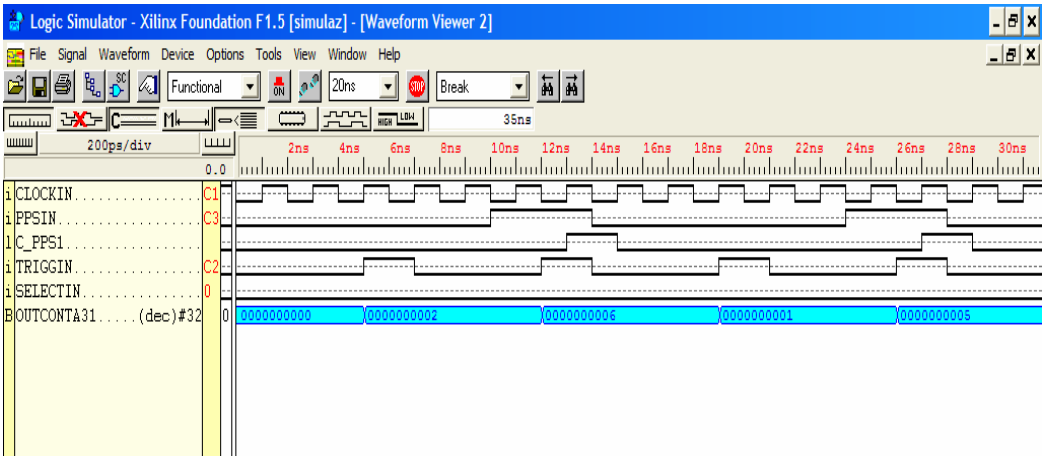


Fig. 4.5 – Simulazione del blocco "contaclock"

L'uscita indicata in figura 4.5 con il nome "outconta31", rappresenta un vettore di 32 bit, visualizzato in decimale. In realtà sarebbero bastati 26 bit per visualizzare l'array di uscita, ma il bus del nostro circuito di test ne possiede 32, ed in questo modo i restanti 6 bit vengono posti a zero.

La simulazione del blocco "contapps" (Fig. 4.6) prevede un array in ingresso, contenente i secondi dall'inizio dell'anno che devono essere sommati al valore del contatore ogni volta che si verifica un evento di segnale di "trigger" attivo alto. Il segnale "selectin" questa volta è posto sempre alto per avere il uscita il valore del contatore di pps, prima invece era al valore logico "0" per visualizzare il conteggio del clock del quarzo.

In questo caso il clock di riferimento è rappresentato dal segnale "pps". All'ingresso "offsetin" è stato assegnato un valore costante uguale a "16" in notazione decimale, quindi ogni volta che il contatore viene azzerato, il suo conteggio riparte dal valore "16". Si notano anche qui una serie di ritardi dovuti al fatto che il segnale "enablein" deve essere normalizzato al segnale "pps" tramite il

"debounce". In figura è anche indicato il valore applicato ad ogni segnale di ingresso, attraverso lo "stimolatore". Nel riquadro "set formula" di figura 4.6 sono indicati nel settore clocks, tre funzioni associate a C1, C2, C3. queste sono associate rispettivamente ai segnali "ppsin", "triggin", "enablein".

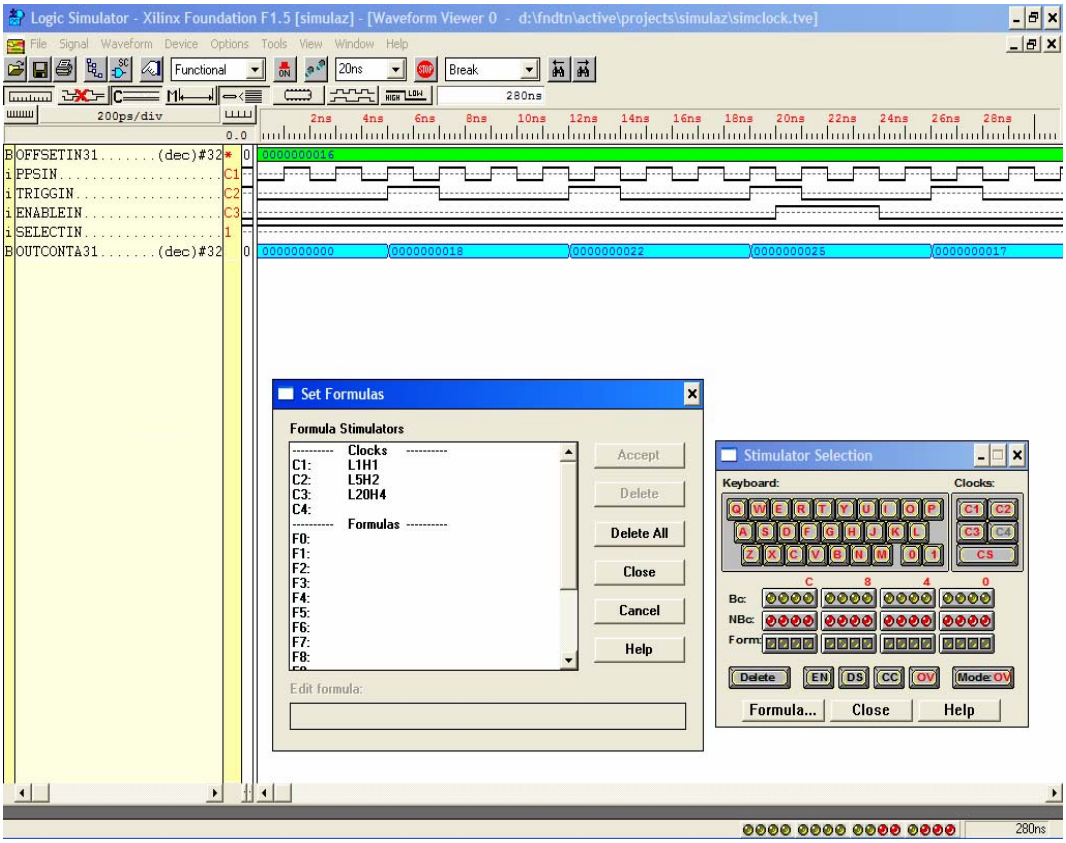


Fig. 4.6 – Simulazione del blocco "contapps" e visualizzazione degli stimolatori applicati.

Una volta verificato il comportamento logico del circuito implementato, la fase successiva consiste nella realizzazione circuitale e la programmazione del dispositivo FPGA. Nel prossimo paragrafo saranno indicati in dettaglio tutti i passaggi della fase di realizzazione del circuito.

4.5. Realizzazione circuitale

Il circuito per verificare il funzionamento della SPARTAN-XL è relativamente semplice (Fig. 4.7). Esso è costituito dalle seguenti parti:

- uno stadio di alimentazione del circuito che, alimentato con valori di tensione maggiori o uguali di +15V fornisce oltre a questa, una tensione di +5V per la logica e di +3.3V per l'FPGA;
- due oscillatori al quarzo collegati ad uno zoccolo che in base a come è connesso permette l'uso o esclude il quarzo non stabilizzato;
- un connettore collegato ai pin di ingresso/uscita della FPGA usati preferibilmente per i segnali di clock;
- una memoria EEPROM necessaria per programmare la SPARTAN-XL;
- otto connettori, collegati alle linee di ingresso/uscita standard del dispositivo.

La FPGA è saldata in modo permanente sul circuito stampato, mentre la memoria EEPROM è connessa al circuito tramite uno zoccolo. Questo permette di poter estrarre facilmente la memoria e programmarla tramite un opportuno dispositivo collegato al PC.

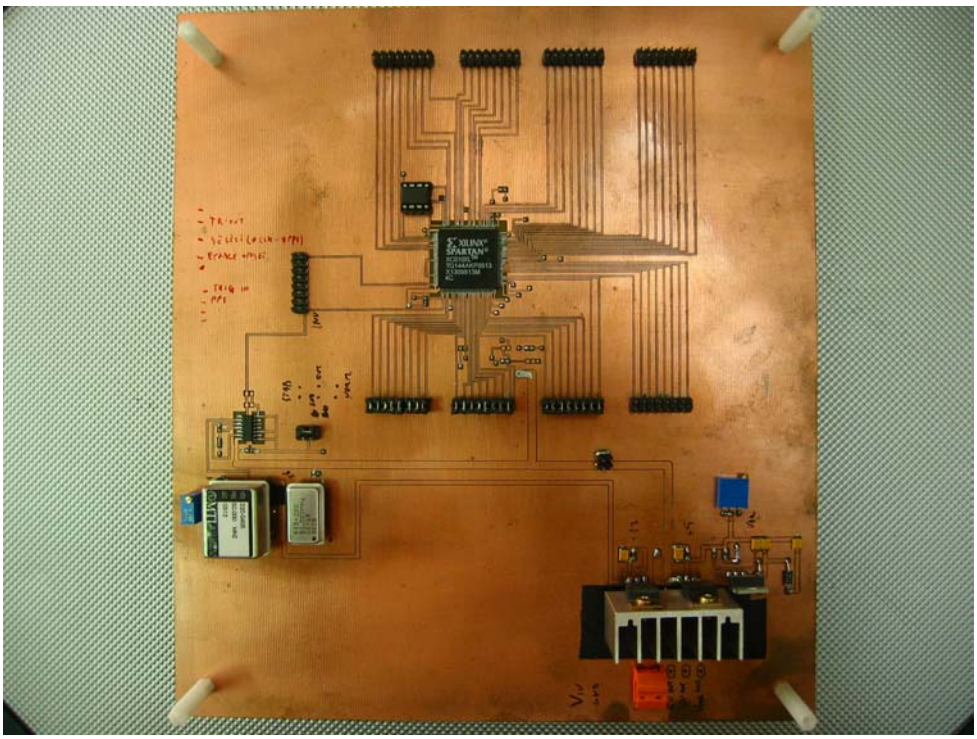


Fig.4.7 – Circuito di test per FPGA

Prima di programmare la EEPROM bisogna implementare il codice scritto. L'implementazione prevede una serie di operazioni tra le quali il passaggio tra descrizione comportamentale e descrizione a porte logiche; di conseguenza il passo successivo deve essere la mappatura del circuito, cioè l'assegnazione reale ad ogni segnale di un piedino fisico sul dispositivo. Questa associazione avviene grazie al fatto che il software include il datasheet della SPARTAN-XL utilizzata e grazie alla modifica di un file con estensione "ucf" nel quale bisogna inserire i nomi dei segnali e i rispettivi pin che si vogliono associare. Se questa operazione non viene effettuata la mappatura avviene comunque, ma in modo casuale. Un esempio di assegnazione di un segnale ad un determinato pin è il seguente:

```
NET clockIN LOC = P2;
NET ppsIN LOC = P39;
NET triggIN LOC= P70;
NET selectIN LOC=P106;
NET enableIN LOC=P76;
```

I segnali, vengono indicati con il loro nome, mentre il nome del pin è ottenuto dal datasheet della SPARTAN-XL. Ogni pin della FPGA è collegato al bus di ingresso/uscita oppure direttamente dal circuito stampato ad uno dei componenti del circuito. Lo schema elettrico è riportato in figura 4.8. Una visione completa del file contenente la mappatura dei segnali si può trovare nell'appendice D.

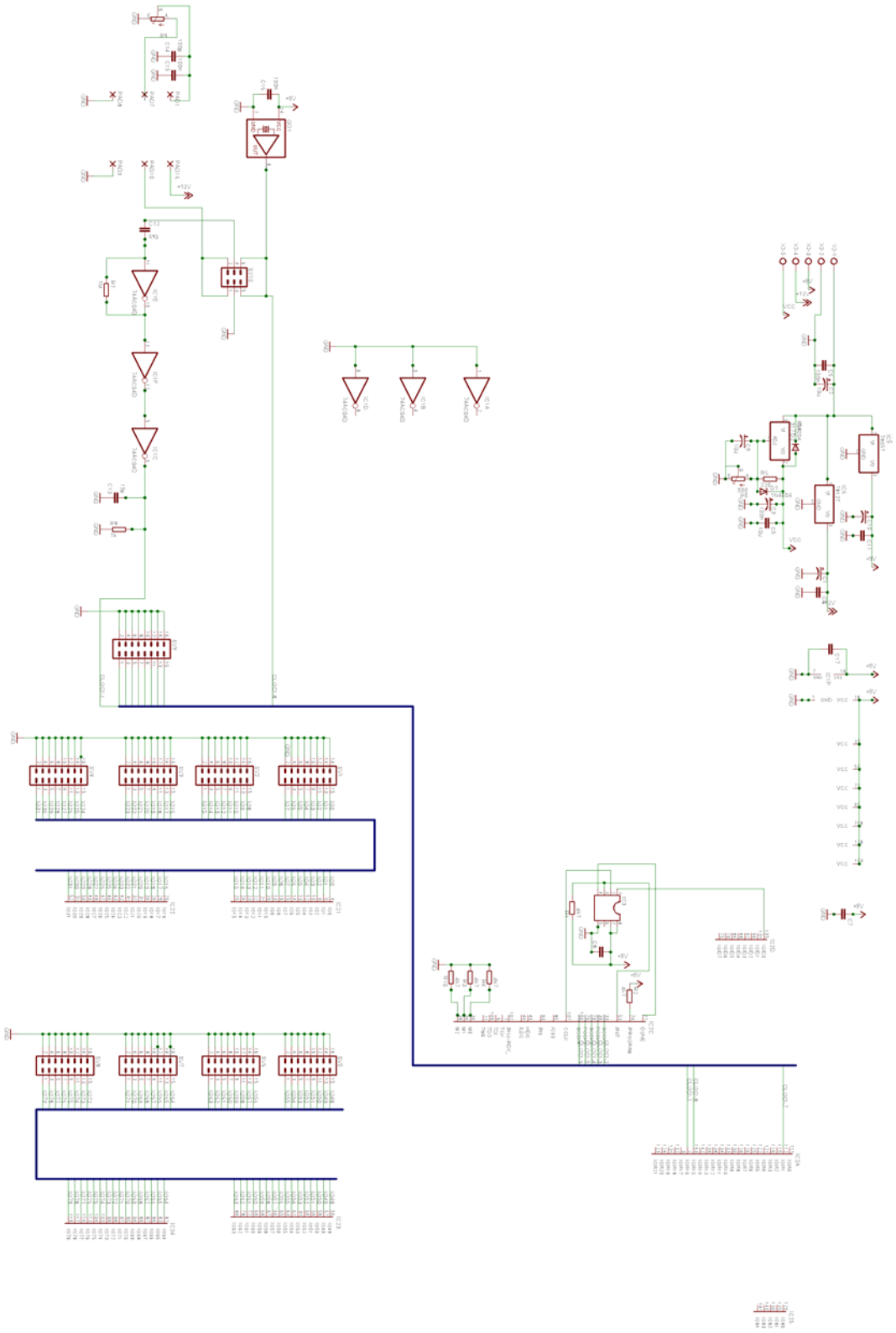


Fig.4.8 – Schema elettrico del circuito per FPGA

Gli ingressi e le uscite sono così disposte: il bus a 32 pin più vicino allo stadio di alimentazione corrisponde all'ingresso dell'offset proveniente dalla PIC; il secondo bus a 32 bit corrisponde all'uscita di uno dei due contatori implementati in base al valore logico del segnale "select". Tutti gli altri segnali di ingresso/uscita del contatore sono situati sul bus a 8 bit singolo collegato alle porte di in/out dell'FPGA preferenziali per i segnali di clock (fig. 4.9).

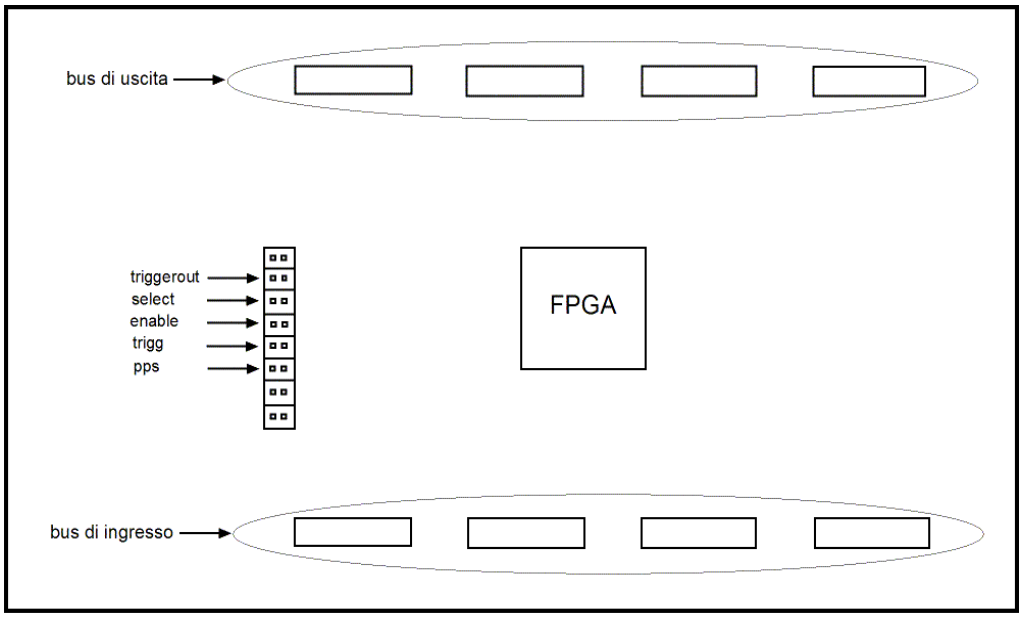


Fig.4.9 – Disposizione dei segnali di ingresso/uscita sui bus del circuito

Una volta realizzato il circuito e stabilita la corrispondenza di ogni segnale con un pin fisico, si procede effettuando il test di funzionamento e l'analisi dei valori ottenuti.

4.6. Test del dispositivo

Per il test del dispositivo si è utilizzato un data generator della *Tektronix* (DG2020A) il quale, sincronizzato con il segnale "pps" proveniente dal ricevitore GPS, fornisce in ingresso al circuito i segnali "pps" e "trigger", permettendo di impostare correttamente la durata dello stato alto dei segnali, la frequenza e lo sfasamento tra i due visualizzandoli sul display. L'uscita del circuito di test è collegata in ingresso ad un analizzatore di stati logici della *Tektronix* (TLA715)

che visualizza l'andamento del segnale e permette di effettuare una statistica sulla stabilità dei valori ottenuti. Questo primo approccio con segnali in ingresso ideali è necessario per verificare il corretto funzionamento del dispositivo; in un secondo momento si utilizzeranno ingressi reali che sono comunque segnali molto stabili e uguali ai precedenti.

4.6.1. Test del contatore "contaclock"

In figura 4.10 è riportata l'uscita del contatore realizzato, corrispondente ad un segnale "select" basso e quindi ad un avanzamento del contatore relativo al blocco "contaclock".

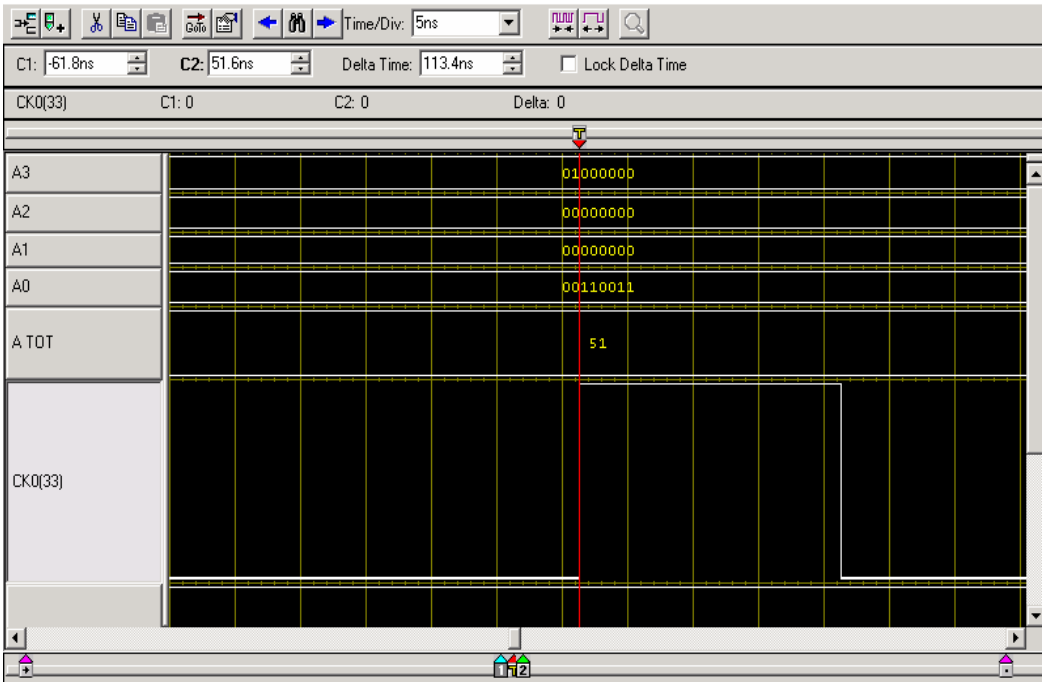


Fig.4.10 – Segnale di uscita del contatore, quando è selezionato il "contaclock"

Il segnale ATOT rappresenta l'uscita del contatore mentre il segnale CK0 rappresenta il segnale "triggerout" prelevato dal circuito e corrispondente al trigger in ingresso una volta applicato il "debounce". L'uscita del contatore è l'insieme dei bit dei quattro bus di uscita del circuito, esclusi i 3 bit più

significativi del bus A3. I 29 bit disponibili in uscita (considerando il bit più a destra in figura 4.9 quello meno significativo) convertiti in decimale, rappresentano il valore indicato in ATOT. Per verificare l'esattezza del valore ottenuto in uscita, si assegnano ai segnali di ingresso "pps" e "trigger" dei valori noti che permettono il calcolo del valore di uscita graficamente, attraverso il data generator (Fig. 4.11).

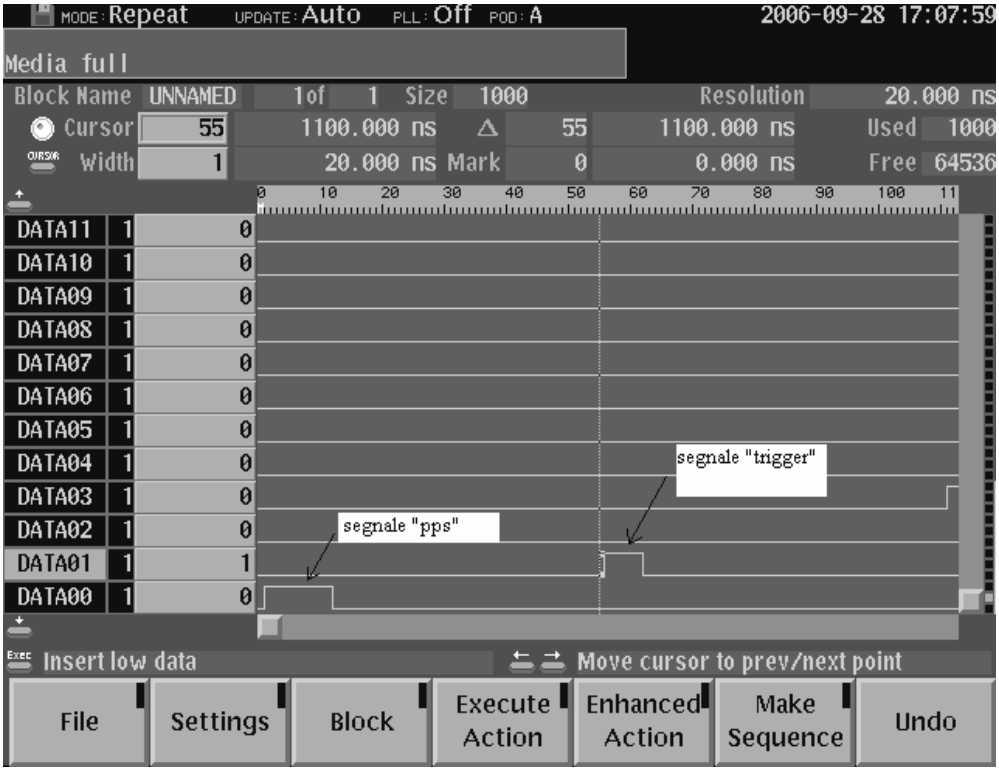


Fig.4.11 – Segnali generati dal data generator ed inviati al circuito come ingressi: partendo dal basso il primo segnale rappresenta il "pps" il secondo il "trigger"

Considerando il fatto che il segnale in ingresso "pps" viene normalizzato, introducendo un ritardo sul fronte di salita reale del "pps" di circa tre passi (ogni passo in figura rappresenta un intervallo di 20 nanosecondi), si può calcolare, a partire dai segnali riportati in figura 4.11, la distanza tra i due fronti di salita dei segnali "pps" e "trigger". Il primo segnale in basso rappresenta il "pps". Il suo fronte di salita è posto nella scala del data generator al valore 1 che in realtà, dopo la normalizzazione, si posiziona al valore quattro. Il secondo segnale rappresenta

il "trigger", che ha il fronte di salita posizionato al valore 55 (campo "cursor" in figura 4.11). Il segnale in uscita al contatore deve essere la differenza tra questi due valori che è pari a 51 e corrisponde esattamente al valore dell'uscita del contatore riportata in figura 4.10. La sequenza dei segnali "pps" e "trigger" è pilotata da un segnale di trigger in ingresso al data generator rappresentato dal "pps" reale proveniente dal ricevitore GPS, in questo modo il "pps" generato dal data generator risulta sincronizzato con il segnale pps reale.

Il valore di uscita dovrebbe essere costante visto che la distanza tra i due fronti di salita degli ingressi è fissata dal generatore di segnale. In realtà osservando l'analizzatore di stati logici, si nota di tanto in tanto un cambiamento del valore di uscita dovuto al fatto che tutta la strumentazione lavora al limite della sua risoluzione. L'analizzatore di stati logici è in grado di effettuare una statistica sull'andamento del segnale di uscita, visualizzando un istogramma nel quale si osserva la percentuale dei valori rivelati (fig.4.12). Nel caso in esame, al dato 51 è associato un valore pari a 88.89% mentre il restante 11.11% delle rivelazioni è distribuito sul clock precedente e su quello sucessivo.

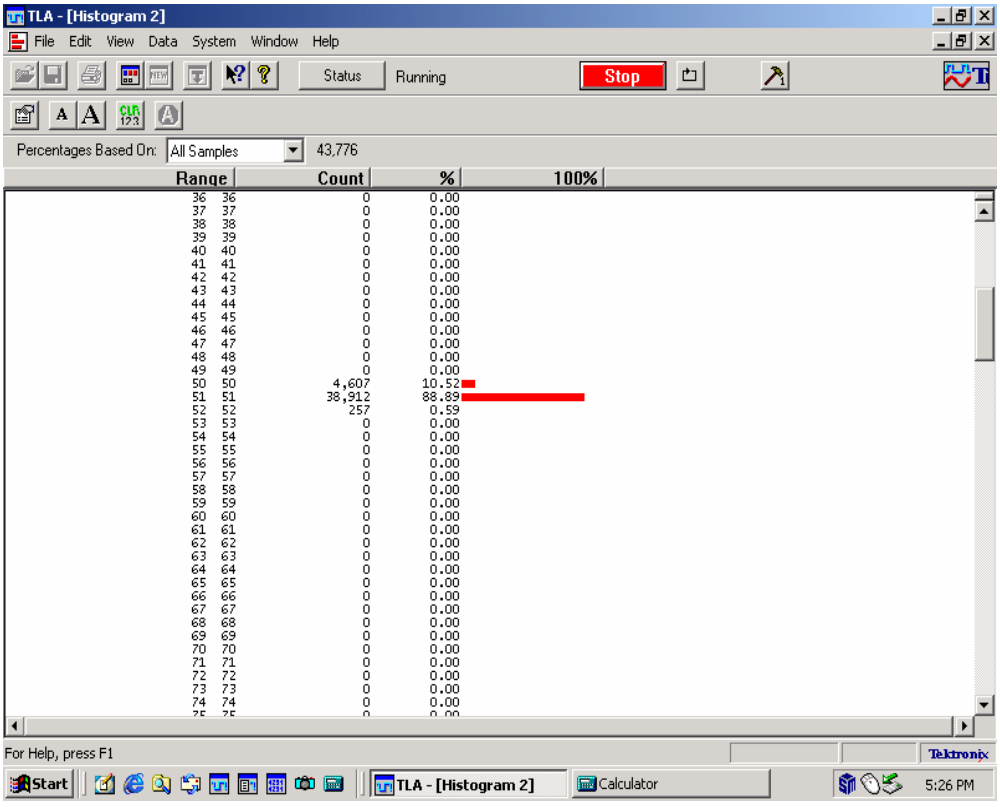


Fig. 4.12 – Istogramma dei dati acquisiti con il logic analyzer

Il logic analyzer è impostato per acquisire 256 campioni (bin) del dato ad ogni impulso di trigger ricevuto. Nel caso in esame, accanto al valore 51 è indicato un conteggio pari a 38.912 mentre in alto si ha un valore pari a 43.776. Questo vuol dire che su 43.776 campioni, 38.912 sono posizionati al valore 51 mentre i restanti sono distribuiti sui 2 valori primi vicini. Quindi su 171 (43.776/256) impulsi di trigger, 152 hanno individuato il valore 51.

I risultati ottenuti circa la precisione del dato in uscita possono essere ritenuti validi per l'applicazione in esame ma un'ulteriore prova sul circuito ci permette di stabilire la fonte dell'errore riscontrato. L'oscillatore interno al data generator ha una frequenza massima di 200MHz ed è inoltre stabilizzato in temperatura. Attraverso un divisore questa frequenza viene portata al valore di 50MHz in modo che sia sincrona con la frequenza del quarzo che fornisce gli impulsi di clock al contatore. In realtà la precisione dei due quarzi in esame è differente infatti, nel primo ha un valore pari a 50 ppm (parti per milione), nel secondo il valore è di 7 ppb (parti per miliardo). È evidente che il quarzo del circuito è molto più preciso del quarzo interno allo strumento, per questo non saranno mai perfettamente sincronizzati. Per verificare l'influenza di questo sfasamento sulla statistica dei dati in uscita al contatore, si è implementato lo schema in figura 4.13.

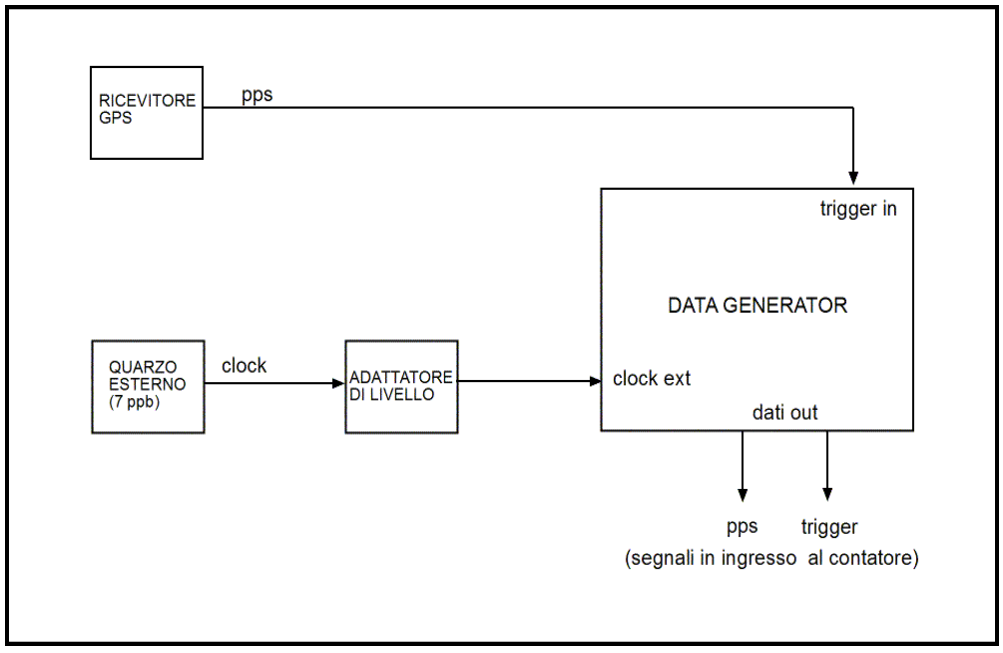


Fig. 4.13 – schema a blocchi del circuito per il quarzo esterno

Il data generator permette di usare un clock esterno in alternativa a quello disponibile al suo interno, per questo si è utilizzato il quarzo installato sul circuito del contatore come clock unico per il sistema fornito in ingresso al data generator tramite un adattatore di livello che converte il segnale da TTL in logica 0 – 2 V. In questo modo i segnali in ingresso al contatore "pps" e "trigger" sono esattamente sincronizzati dall'unico segnale di clock fornito dal quarzo del circuito. Un'analisi dell'istogramma dei dati in uscita (fig. 4.14), mostra che: tutti i dati prelevati dal contatore sono distribuiti su di un unico valore, pertanto l'errore riscontrato in precedenza è dovuto alla precisione del clock interno del data generator.

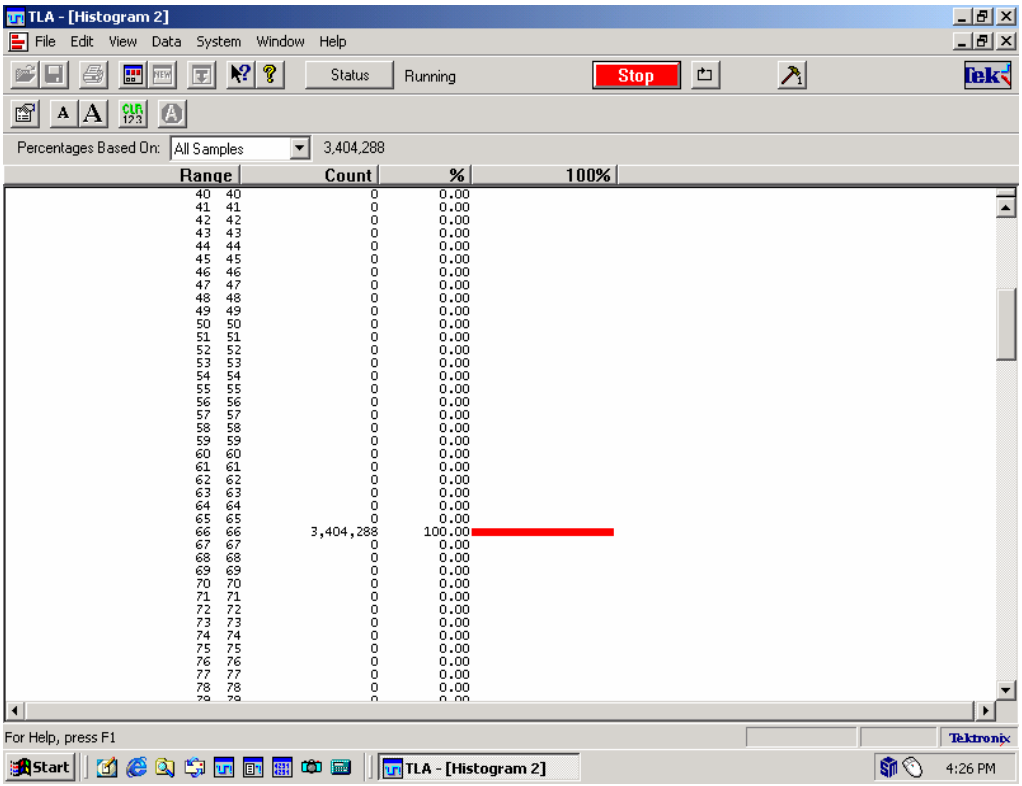


Fig. 4.14 – Istogramma dei dati acquisiti con il logic analyzer e uso di un unico clock

Per i motivi precedentemente esposti, nel circuito finale il quarzo usato non è stabilizzato, per questo si è voluta verificare la distribuzione dei dati sull'istogramma fornendo al circuito il clock del quarzo non stabilizzato mentre, al data generator, un clock esterno dato dal quarzo stabilizzato. Il risultato è una distribuzione delle acquisizioni su valori che si discostano da quello centrale anche di 25 cicli di clock (fig. 4.15). Il valore 51 è stato acquisito per circa il 90% del tempo su un intervallo di circa un'ora. Si osserva quindi che il quarzo non stabilizzato introduce un errore sulla precisione dei dati in uscita comunque superiore a quello di figura 4.12 che riteniamo accettabile.

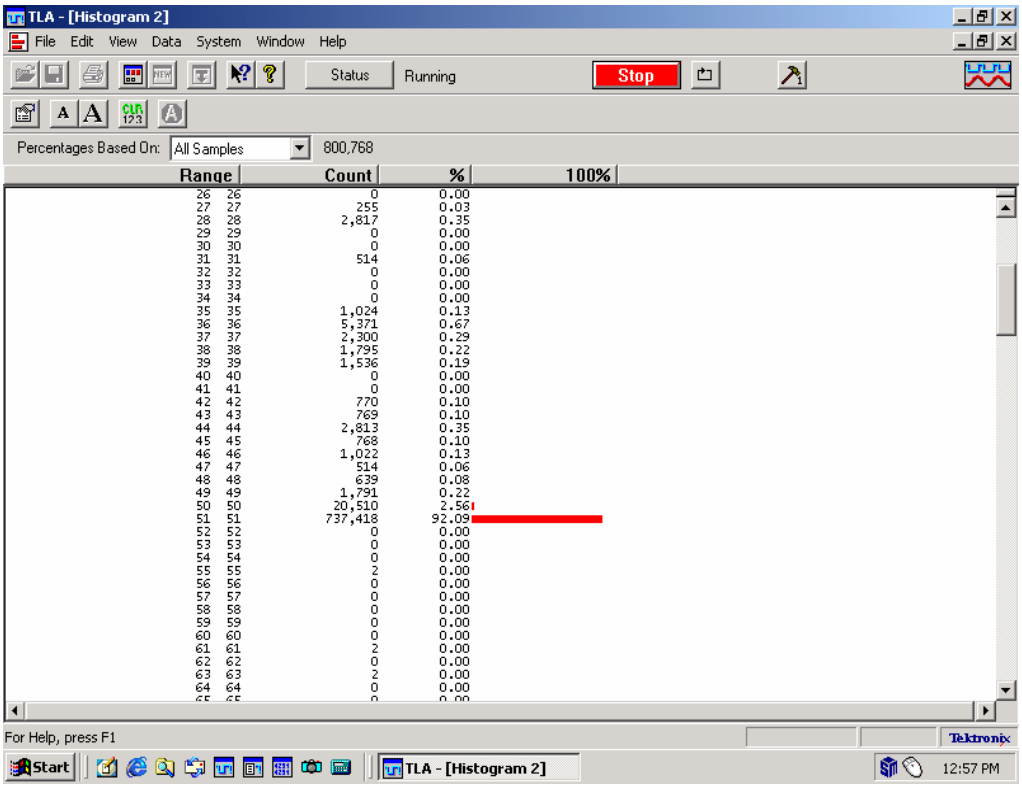


Fig. 4.15 – Istogramma dei dati acquisiti con il logic analyzer (clock non stabilizzato)

4.6.2. Test del contatore "contapps"

Assegnando al segnale "select" un valore logico alto si seleziona il contatore del segnale pps.

Il contatore deve acquisire i secondi trascorsi dall'inizio dell'anno tramite il segnale "offset", nell'istante in cui il segnale "enable" proveniente dalla PIC avvisa che i dati in uscita sono attendibili, inoltre incrementa il suo valore di un passo a partire dall'offset ottenuto con impulsi di clock dati dal segnale "pps" proveniente dalla PIC. In pratica il segnale di uscita del contatore deve corrispondere esattamente all'avanzamento dei secondi dall'inizio dell'anno forniti in uscita dalla PIC.

Prima di effettuare il test, facciamo un passo indietro sulle istruzioni del firmware per comprendere meglio il funzionamento del contatore. La logica di avanzamento del contatore include, come già visto, due funzioni, una di somma di due numeri binari e una di incremento di un numero binario. Considerando come segnali di ingresso "pps" e "trigger" quelli usati per il "contaclock" (fig. 4.11), bisogna definire il segnale "enable". Dal codice del firmware si può vedere che un contatore incrementa il suo valore di un passo ad ogni impulso del segnale "pps" finchè il segnale "enable è mantenuto al valore logico zero. Quando il segnale "enable" subisce una modifica passando dallo stato logico basso a quello alto, nell'istante in cui si ha il fronte di salita del segnale una variabile interna memorizza il valore del segnale "offset" proveniente dalla PIC. Il contatore continua ad essere azzerato ad ogni impulso di pps, finchè il segnale "enable" rimane nello stato alto. Quando "enable" torna nello stato basso il contatore ricomincia il suo avanzamento. Ad ogni impulso di "trigger", infine, il valore del contatore viene sommato al valore dell'offset memorizzato precedentemente e il risultato rappresenta l'uscita del "contapps". Da questo si capisce che l'andamento ideale del segnale "enable" deve essere impulsivo, infatti questo segnale proveniente dalla PIC ha una durata dello stato alto dell'ordine dei microsecondi.

Per l'analisi dell'uscita del "contapps" si pone, in un primo momento, il bus di ingresso del segnale "offset" (fig. 4.9) ad un valore prestabilito pari a 14 in decimale collegando a massa i pin che devono essere posti ad un valore logico

pari a zero e collegando alla tensione +5V quelli che devono assumere uno stato logico alto.

Attraverso l'assegnazione di un offset in ingresso direttamente dalla porta del circuito con un valore conosciuto a priori, si verifica, portando il segnale "enable" ad uno stato alto in modo permanente, che l'uscita del contatore collegata al logic analyzer assume il valore dell'offset inserito; infatti nell'istante in cui il segnale "enable" è portato alto, il dispositivo preleva il valore di offset inserito in ingresso e azzera il contatore in continuazione, visto che il segnale "enable" rimane al valore alto. La somma del valore dell'offset con quello del contatore, che è sempre nullo, restituisce in uscita l'offset applicato in ingresso. Una volta verificata la perfetta acquisizione dell'offset, si procede applicando il segnale di "offset" proveniente dalla PIC e il segnale "enable" fornito dal data generator (segnale DATA02 in fig. 4.16). In questo caso il valore dell'offset in ingresso al dispositivo non è costante e quindi l'uscita del contatore, quando è applicato un impulso del segnale "enable" a passi di "pps", corrisponde esattamente all'avanzamento dell'offset in ingresso. In questa configurazione il contatore non fa altro che riportare in uscita il valore di ingresso che corrisponde all'avanzamento dei secondi dall'inizio dell'anno della PIC (fig. 4.17).

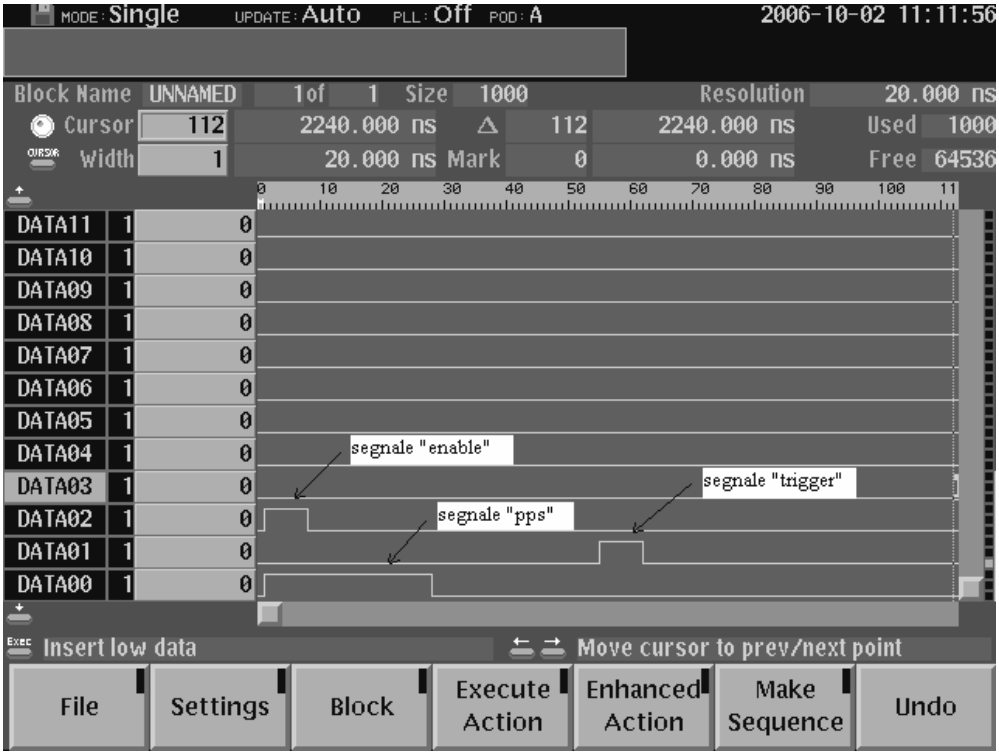


Fig.4.16 – Segnali generati dal data generator: dal basso il primo segnale rappresenta il "pps" il secondo il "trigger" ed il terzo "enable"

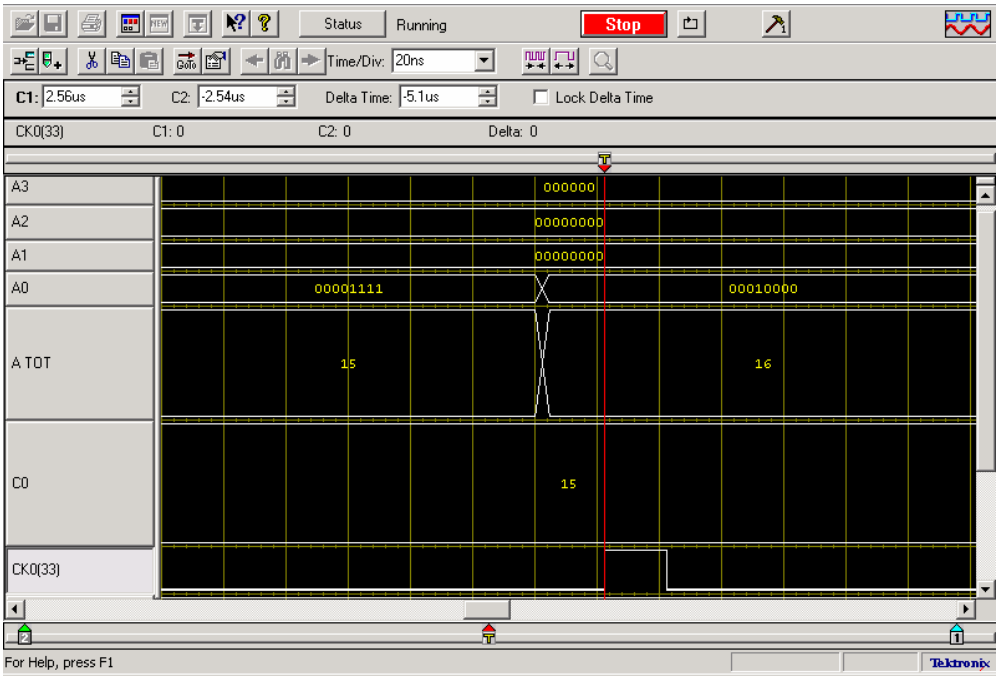


Fig.4.17 – Segnale di uscita del "contapps" con offset dalla PIC e segnale "enable"

In questo caso il segnale ATOT corrisponde all'uscita del "contapps" che avanza ad ogni fronte di salita del segnale "trigger" rappresentato in figura dal segnale CK0 mentre "C0" corrisponde al segnale "offset" proveniente dalla PIC.
L'obiettivo del contatore, però, è quello di ottenere il conteggio dei secondi dall'inizio dell'anno in modo indipendente dalla PIC. L'unico compito di quest'ultima è quello di fornire la base del conteggio che procederà successivamente grazie al segnale "pps" del ricevitore. La verifica finale è stata quella di porre il segnale "enable" in uno stato logico basso dopo aver acquisito l'offset in ingresso, in questo stato l'uscita del contatore è indipendente dall' ingresso di offset, ma, come previsto, assume lo stesso valore.

CONCLUSIONI E SVILUPPI FUTURI

L'obiettivo di ottenere la sincronizzazione temporale tra sistemi di acquisizione remoti con relativo time stamping dei dati dislocati è stato raggiunto con successo attraverso l'uso di dispositivi adatti alle esigenze da espletare.
I punti fondamentali da sviluppare sono stati:

- scelta di un dispositivo di sincronizzazione temporale dei dati ad alta precisione;
- estrazione dei dati necessari dal dispositivo per ottenere riferimenti di posizione e tempo universali;
- realizzazione di un dispositivo sincronizzato con il riferimento temporale per la gestione degli eventi con precisione dell'ordine dei nanosecondi;
- acquisizione dei dati e test della risoluzione ottenuta dal dispositivo finale.

Il dispositivo utilizzato per ottenere un riferimento temporale di alta precisione e disponibile in qualsiasi punto territoriale è il GPS, accessibile tramite un ricevitore GPS che utilizza un protocollo dei dati chiamato TSIP il quale permette, tramite il calcolo dell'errore di quantizzazione, la massima precisione nella ricezione del segnale "pps" corrispondente ad un impulso ogni secondo. L'estrazione delle informazioni necessarie per il time stamping dei dati è stata possibile grazie all'uso di una PIC che implementa un firmware appositamente creato per ottenere la posizione geografica degli strumenti ed il riferimento temporale espresso in secondi dall'inizio dell'anno. La parte più complessa del progetto relativa al terzo punto è stata risolta attraverso l'uso di una FPGA e di un quarzo da 50MHz che permettono di ricevere il dato relativo ai secondi dall'inizio dell'anno dalla PIC e incrementare un contatore con risoluzione dell'ordine del nanosecondo che riparte ad ogni impulso del segnale "pps" e permette il time stamping sui dati ogni volta che un impulso di trigger segnala la presenza di un dato rivelato. Accanto all'informazione dell'evento avvenuto al determinato istante è disponibile anche l'informazione dei secondi dall'inizio dell'anno, fornita però non dalla PIC ma dalla stessa FPGA che riceve solo inizialmente questa informazione dalla PIC e la incrementa in corrispondenza di ogni impulso

di “pps” ricevuto dal ricevitore GPS. L’acquisizione dei dati temporali con il dispositivo finale fornisce una statistica sulla precisione di quest’ultimo: l’utilizzo di un quarzo stabilizzato in temperatura ad alta precisione permette di ottenere la totale distribuzione dei valori rivelati per lunghi periodi, sul determinato impulso di clock. L’uso di un quarzo non stabilizzato introduce invece un significativo errore di sincronizzazione nei dati.

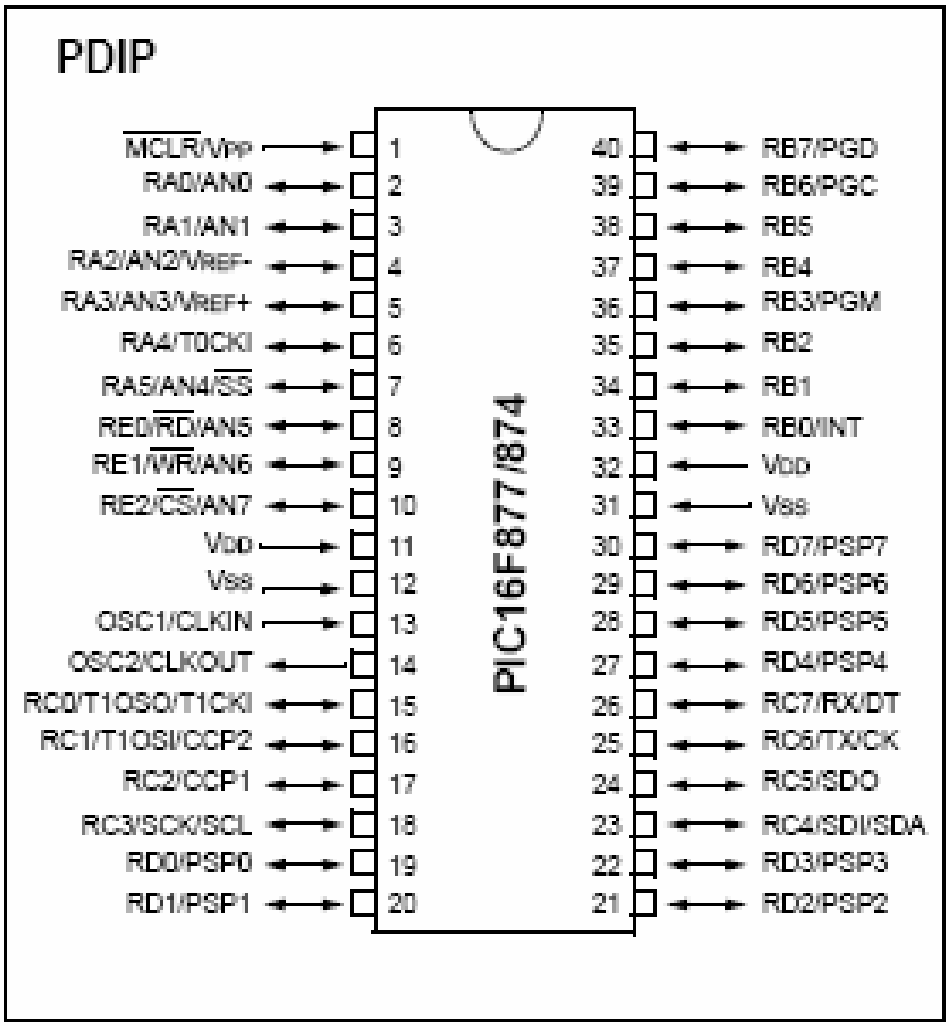
Gli sviluppi futuri del dispositivo riguardano l’integrazione di quest’ultimo su di un modulo VME in modo da renderlo disponibile in diverse applicazioni. Bisogna inoltre implementare un registro, interno o esterno all’FPGA, per la raccolta delle informazioni prelevate da questa; in particolare, ad ogni evento rivelato dal telescopio in corrispondenza di un segnale di trigger bisogna associare i seguenti valori:

- riferimento temporale con precisione dell’ordine del nanosecondo;
- errore di quantizzazione relativo al segnale pps;
- numero di cicli di clock compiuti dal quarzo non stabilizzato nell’intervallo di segnale pps considerato.

Ognuno di questi dati è disponibile però in momenti differenti, per questo, oltre a stabilire il metodo migliore per la registrazione dei dati, occorre anche prelevare il dato nell’istante giusto considerando i ritardi del dispositivo da cui l’informazione viene estratta. Un’ulteriore implementazione riguarda la normalizzazione dei cicli di clock effettuati dal quarzo ad ogni segnale pps, cioè ottenere, attraverso una proporzione, il numero di cicli di clock del quarzo in corrispondenza di un segnale di trigger, supponendo che il numero di oscillazioni nell’intervallo del pps siano esattamente 50.000.000. Questa operazione non può essere effettuata da una FPGA perché essa non è in grado di operare con moltiplicazioni e divisioni che non siano per due. Una strada possibile sarebbe quella di ricondurre moltiplicazioni e divisioni a semplici operazioni di somma e sottrazione attraverso l’uso di un algoritmo appropriato.

APPENDICE A

PIEDINATURA DELLA PIC



CARATTERISTICHE PRINCIPALI

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F877
Operating Frequency	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory	256
Interrupts	14
I/O Ports	Ports A,B,C,D,E
Timers	3
Capture/Compare/PWM Modules	2
Serial Communications	MSSP, USART
Parallel Communications	PSP
10-bit Analog-to-Digital Module	8 input channels
Instruction Set	35 instructions

APPENDICE B

LISTATO FIRMWARE DELLA PIC

Di seguito è riportato il listato del firmware della PIC. Le righe di codice evidenziate in rosso rappresentano una versione precedente, queste sostituivano la riga successiva. Tutte le informazioni prelevate dal GPS sono contenute nei vettori DATAOUT e DATAOUT2 che corrispondono rispettivamente ai pacchetti ricevuti 10x8F-AB e 10x8F-AC. Per estrapolare dall'array un determinato dato, si ottiene l'indice facendo riferimento al numero del byte del dato voluto indicato nella tabella del rispettivo pacchetto, sottratto di una unità.

```
'*****************************************************************
'*  Name    : PICFIRMWARE.BAS
'*  Author  : ANDREA DONNO
'*  Date    : 03/05/2006
'*  Version : 1.0
'*****************************************************************

DEFINE OSC 20
  Define ONINT_USED 1
  include "modedefs.bas"

  ADCON1=7
   TRISC=%10000000
   TRISA=0      'ABILITA IL PORTA COME USCITA
   TRISD=0              'ABILITA IL PORTD COME USCITA


  'ISTRUZIONI PER LA COMUNICAZIONE CON IL DISPLAY LCD
  DEFINE LCD_DREG   PORTB
  DEFINE LCD_DBIT 4
  DEFINE LCD_BITS 4
  Define LCD_RSREG  PORTB
  DEFINE LCD_RSBIT 1
  DEFINE LCD_EREG   PORTB
  DEFINE LCD_EBIT 3
  DEFINE LCD_LINES 2
  DEFINE SER2_ODD 1
  DEFINE SER2_BITS 9
  ORE VAR BYTE
  S VAR BYTE
  M VAR BYTE
  H VAR BYTE
  D VAR BYTE
```

```
MESE VAR BYTE
A1 VAR byte
A2 VAR byte
ANNO VAR WORD
S1 VAR BYTE
S2 VAR BYTE
M1 VAR BYTE
H1 VAR BYTE
D1 VAR BYTE
MESE1 VAR BYTE
ANNO1 VAR WORD
NSAT VAR BYTE
SAT VAR BYTE
DATIX VAR BYTE [22]
DATIIN VAR BYTE[18]
DATIIN2 VAR BYTE[67]
DATIOUT VAR BYTE[18]
DATIOUT2 VAR BYTE[67]
EQ1 VAR BYTE[4]
I VAR BYTE
J VAR BYTE
A VAR BYTE
B VAR BYTE
C VAR BYTE
E VAR BYTE
CONTROLLOSAT VAR PORTC.5
SECANNOH VAR WORD
SECANNOL VAR WORD
SECANNOLINV VAR WORD
SECANNO VAR WORD
DAYANNO VAR WORD
N VAR BYTE
TEMP VAR WORD
USCITA2      VAR PORTD
LCDOUT $FE, 1, "DATI PROVENIENTI"
LCDOUT $FE, $C0, "DAL GPS"

INI:
A=0
I=0
C=0
J=0
E=0
SERIN2 PORTC.7,8276,[WAIT ($108F),WAIT ($AB), STR DATIIN\18 ]
SERIN2 PORTC.7,8276,[WAIT ($108F),WAIT ($AC), STR DATIIN2\67 ]

'SERIN2 PORTC.7,8276,[WAIT ($8FAB),SKIP 9,S,M,H,D,MESE, a1,a2 ]
FOR I=0 TO 17
IF (DATIIN[I] == $10) AND (DATIIN[I+1] == $10 )THEN
FOR B=I TO 16
DATIIN[B]=DATIIN[B+1]
NEXT B
DATIOUT[A]=DATIIN[I]
A=A+1
ELSE
DATIOUT[A]=DATIIN[I]
A=A+1
ENDIF
NEXT I

FOR I=0 TO 66
IF (DATIIN2[I] == $10) AND (DATIIN2[I+1] == $10 )THEN
```

```
FOR B=I TO 65
DATIIN2[B]=DATIIN2[B+1]
NEXT B
DATIOUT2[C]=DATIIN2[I]
C=C+1
ELSE
DATIOUT2[C]=DATIIN2[I]
C=C+1
ENDIF
NEXT I


ANNO.byte1 = DATIOUT[14]
ANNO.byte0 = DATIOUT[15]
EQ1[0]=DATIOUT2[59]
EQ1[1]=DATIOUT2[60]
EQ1[2]=DATIOUT2[61]
EQ1[3]=DATIOUT2[62]

FOR J=0 TO 40
IF  (ANNO != 2008+4*J) THEN
N=0
ELSE
N=1
GOTO DATI
ENDIF
NEXT J

DATI:
IF  DATIOUT[13]== 1    THEN   DAYANNO = DATIOUT[12]
IF  DATIOUT[13]== 2    THEN   DAYANNO = DATIOUT[12]+31
IF  DATIOUT[13]== 3    THEN   DAYANNO = DATIOUT[12] +59+N
IF  DATIOUT[13]== 4    THEN   DAYANNO = DATIOUT[12] +90+N
IF  DATIOUT[13]== 5    THEN   DAYANNO = DATIOUT[12] +120+N
IF  DATIOUT[13]== 6    THEN   DAYANNO = DATIOUT[12] +151+N
IF  DATIOUT[13]== 7    THEN   DAYANNO = DATIOUT[12] +181+N
IF  DATIOUT[13]== 8    THEN   DAYANNO = DATIOUT[12] +212+N
IF  DATIOUT[13]== 9    THEN   DAYANNO = DATIOUT[12] +243+N
IF  DATIOUT[13]== 10   THEN   DAYANNO = DATIOUT[12] +273+N
IF  DATIOUT[13]== 11   THEN   DAYANNO = DATIOUT[12] +304+N
IF  DATIOUT[13]== 12   THEN
DAYANNO = DATIOUT[12] +334
  ENDIF

TEMP= DATIOUT[11] + (24*DAYANNO)
SECANNOH = 3600**TEMP
SECANNOL = 3600*TEMP
          'MINUTI      'SECONDI
SECANNO = 60* DATIOUT[10]+ DATIOUT[9]
IF (65535 - SECANNOL) <= SECANNO THEN
SECANNOH=SECANNOH+1
ENDIF
SECANNOL=SECANNOL+SECANNO


LCDOUT $FE, 1, dec DATIOUT[12],"/", dec DATIOUT[13],"/",dec ANNO ,      " ",dec
DATIOUT[11],":",dec DATIOUT[10],":",dec DATIOUT[9]
SEROUT2 PORTC.6,8276,[$10,$24,$10,$03]
SERIN2 PORTC.7,8276,[WAIT ($106D), NSAT]
SAT = NSAT >> 4
```

```
  'LCDOUT $FE, $C0,DEC SAT , " EQ:", HEX EQ1[0],HEX EQ1[1],HEX       EQ1[2],HEX
EQ1[3]
  LCDOUT $FE, $C0,"SECANNO:",DEC SECANNOH,DEC SECANNOL

  CONTROLLOSAT=1
  SECANNOLINV=SECANNOL REV 16
  USCITA2=SECANNOLINV.Byte1
  CONTROLLOSAT=0

  GOTO INI


  END
```

APPENDICE C
PIEDINATURA DELL' FPGA

XCS10 and XCS10XL Device Pinouts

XCS10/XL Pad Name	PC84	VQ100	CS144[2]	TQ144	Bndry Scan
VCC	P2	P89	D7	P128	-
I/O	P3	P90	A6	P129	44
I/O	P4	P91	B6	P130	47
I/O	-	P92	C6	P131	50
I/O	-	P93	D6	P132	53
I/O	P5	P94	A5	P133	56
I/O	P6	P95	B5	P134	59
I/O	-	-	C5	P135	62
I/O	-	-	D5	P136	65
GND	-	-	A4	P137	-
I/O	P7	P96	B4	P138	68
I/O	P8	P97	C4	P139	71
I/O	-	-	A3	P140	74
I/O	-	-	B3	P141	77
I/O	P9	P98	C3	P142	80
I/O, SGCK1[1] GCK8[2]	P10	P99	A2	P143	83
VCC	P11	P100	B2	P144	-
GND	P12	P1	A1	P1	-
I/O, PGCK1[1] GCK1[2]	P13	P2	B1	P2	86
I/O	P14	P3	C2	P3	89
I/O	-	-	C1	P4	92
I/O	-	-	D4	P5	95
I/O, TDI	P15	P4	D3	P6	98
I/O, TCK	P16	P5	D2	P7	101
GND	-	-	D1	P8	-
I/O	-	-	E4	P9	104
I/O	-	-	E3	P10	107
I/O, TMS	P17	P6	E2	P11	110
I/O	P18	P7	E1	P12	113
I/O	-	-	F4	P13	116
I/O	-	P8	F3	P14	119
I/O	P19	P9	F2	P15	122
I/O	P20	P10	F1	P16	125
GND	P21	P11	G2	P17	-
VCC	P22	P12	G1	P18	-
I/O	P23	P13	G3	P19	128
I/O	P24	P14	G4	P20	131
I/O	-	P15	H1	P21	134
I/O	-	-	H2	P22	137
I/O	P25	P16	H3	P23	140
I/O	P26	P17	H4	P24	143
I/O	-	-	J1	P25	146
I/O	-	-	J2	P26	149
GND	-	-	J3	P27	-
I/O	P27	P18	J4	P28	152
I/O	-	P19	K1	P29	155
I/O	-	-	K2	P30	158
I/O	-	-	K3	P31	161
I/O	P28	P20	L1	P32	164
I/O, SGCK2[1] GCK2[2]	P29	P21	L2	P33	167
Not Connected[1] M1[2]	P30	P22	L3	P34	170
GND	P31	P23	M1	P35	-
MODE[1], M0[2]	P32	P24	M2	P36	173
VCC	P33	P25	N1	P37	-
Not Connected[1] $\overline{\text{PWRDWN}}$[2]	P34	P26	N2	P38	174 [1]

XCS10/XL Pad Name	PC84	VQ100	CS144[2]	TQ144	Bndry Scan
I/O, PGCK2[1] GCK3[2]	P35	P27	M3	P39	175[3]
I/O (HDC)	P36	P28	N3	P40	178 [3]
I/O	-	-	K4	P41	181 [3]
I/O	-	-	L4	P42	184 [3]
I/O	-	P29	M4	P43	187 [3]
I/O ($\overline{LDC}$)	P37	P30	N4	P44	190 [3]
GND	-	-	K5	P45	-
I/O	-	-	L5	P46	193 [3]
I/O	-	-	M5	P47	196 [3]
I/O	P38	P31	N5	P48	199 [3]
I/O	P39	P32	K6	P49	202 [3]
I/O	-	P33	L6	P50	205 [3]
I/O	-	P34	M6	P51	208 [3]
I/O	P40	P35	N6	P52	211 [3]
I/O ($\overline{INIT}$)	P41	P36	M7	P53	214 [3]
VCC	P42	P37	N7	P54	-
GND	P43	P38	L7	P55	-
I/O	P44	P39	K7	P56	217 [3]
I/O	P45	P40	N8	P57	220 [3]
I/O	-	P41	M8	P58	223 [3]
I/O	-	P42	L8	P59	226 [3]
I/O	P46	P43	K8	P60	229 [3]
I/O	P47	P44	N9	P61	232 [3]
I/O	-	-	M9	P62	235 [3]
I/O	-	-	L9	P63	238 [3]
GND	-	-	K9	P64	-
I/O	P48	P45	N10	P65	241 [3]
I/O	P49	P46	M10	P66	244 [3]
I/O	-	-	L10	P67	247 [3]
I/O	-	-	N11	P68	250 [3]
I/O	P50	P47	M11	P69	253 [3]
I/O, SGCK3[1] GCK4[2]	P51	P48	L11	P70	256 [3]
GND	P52	P49	N12	P71	-
DONE	P53	P50	M12	P72	-
VCC	P54	P51	N13	P73	-
$\overline{PROGRAM}$	P55	P52	M13	P74	-
I/O (D7[2])	P56	P53	L12	P75	259 [3]
I/O, PGCK3[1] GCK5[2]	P57	P54	L13	P76	262 [3]
I/O	-	-	K10	P77	265 [3]
I/O	-	-	K11	P78	268 [3]
I/O (D6[2])	P58	P55	K12	P79	271 [3]
I/O	-	P56	K13	P80	274 [3]

XCS10/XL Pad Name	PC84	VQ100	CS144[2]	TQ144	Bndry Scan
GND	-	-	J10	P81	-
I/O	-	-	J11	P82	277 [3]
I/O	-	-	J12	P83	280 [3]
I/O (D5[2])	P59	P57	J13	P84	283 [3]
I/O	P60	P58	H10	P85	286 [3]
I/O	-	P59	H11	P86	289 [3]
I/O	-	P60	H12	P87	292 [3]
I/O (D4[2])	P61	P61	H13	P88	295 [3]
I/O	P62	P62	G12	P89	298 [3]
VCC	P63	P63	G13	P90	-
GND	P64	P64	G11	P91	-
I/O (D3[2])	P65	P65	G10	P92	301 [3]
I/O	P66	P66	F13	P93	304 [3]
I/O	-	P67	F12	P94	307 [3]
I/O	-	-	F11	P95	310 [3]
I/O (D2[2])	P67	P68	F10	P96	313 [3]
I/O	P68	P69	E13	P97	316 [3]
I/O	-	-	E12	P98	319 [3]
I/O	-	-	E11	P99	322 [3]
GND	-	-	E10	P100	-
I/O (D1[2])	P69	P70	D13	P101	325 [3]
I/O	P70	P71	D12	P102	328 [3]
I/O	-	-	D11	P103	331 [3]
I/O	-	-	C13	P104	334 [3]
I/O (D0[2], DIN)	P71	P72	C12	P105	337 [3]
I/O, SGCK4[1] GCK6[2] (DOUT)	P72	P73	C11	P106	340 [3]
CCLK	P73	P74	B13	P107	-
VCC	P74	P75	B12	P108	-
O, TDO	P75	P76	A13	P109	0
GND	P76	P77	A12	P110	-
I/O	P77	P78	B11	P111	2
I/O, PGCK4[1] GCK7[2]	P78	P79	A11	P112	5
I/O	-	-	D10	P113	8
I/O	-	-	C10	P114	11
I/O (CS1[2])	P79	P80	B10	P115	14
I/O	P80	P81	A10	P116	17
GND	-	-	C9	P118	-
I/O	-	-	B9	P119	20
I/O	-	-	A9	P120	23
I/O	P81	P82	D8	P121	26
I/O	P82	P83	C8	P122	29
I/O	-	P84	B8	P123	32

XCS10 and XCS10XL Device Pinouts

XCS10/XL Pad Name	PC84	VQ100	CS144[2]	TQ144	Bndry Scan
I/O	-	P85	A8	P124	35
I/O	P83	P86	B7	P125	38
I/O	P84	P87	A7	P126	41
GND	P1	P88	C7	P127	-

Notes:

1. 5V Spartan only
2. 3V Spartan-XL only
3. The "PWRDWN" on the XCS10XL is not part of the Boundary Scan chain. For the XCS10XL, subtract 1 from all Boundary Scan numbers from GCK3 on (175 and higher).

Additional XCS10/XL Package Pins

TQ144					
Not Connected Pins					
P117	-	-	-	-	-
5/5/97					

CS144					
Not Connected Pins					
D9	-	-	-	-	-
4/28/99					

APPENDICE D

LISTATO FIRMWARE DELL'FPGA

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

ENTITY debounce IS                    ------debounce
PORT (CLOCK : IN std_logic;
     sign : IN std_logic;
     signout : OUT std_logic);
END debounce;

ARCHITECTURE comp7 OF debounce IS
     signal Q1,Q2,Q3 : std_logic;

BEGIN
process(CLOCK)                        -------
begin

  if (clock'event AND clock = '1') then
    Q1 <= sign;
    Q2 <= Q1;
    Q3 <= Q2;
  end if;
end process;

signout <= Q1 and Q2 and (not Q3);

END comp7;

--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

ENTITY contaclock is
    port (
        pps: in STD_LOGIC;
        clock: in STD_LOGIC;
        trigg: in STD_LOGIC;
        triggout: out STD_LOGIC;
        reset: in STD_LOGIC;
        outA: out STD_LOGIC_VECTOR (31 downto 0));
end contaclock;


ARCHITECTURE contaclock_arch of contaclock is                    --------

signal out1:STD_LOGIC_VECTOR (31 downto 0);
SIGNAL pps1 : std_logic;
```

```
signal debounce_trigg : std_logic;

COMPONENT debounce

PORT (clock : IN std_logic;
     sign : IN std_logic;
     signout : OUT std_logic);

END COMPONENT;

function inc_bv       (a: std_logic_vector) return std_logic_vector is
     variable s: std_logic_vector (31 downto 0);
     variable carry: std_logic;

     begin
               carry := '1';
               for i in 0 to 31 loop
                         s(i) := a(i) xor carry;
                         carry := a(i) and carry;
               end loop;
               return (s);
     end inc_bv;

begin
A : debounce
     port map (clock, pps, pps1);
B : debounce
     port map (clock, trigg, debounce_trigg);

process(clock, pps1)                                                    ---------
begin
if (pps1 = '1') then
     out1 <= "00000000000000000000000000000000";
elsif (clock'event and clock = '1' ) then
     if (out1 > "00000001000011110101111101000000" or reset = '1') then
               out1 <= "00000000000000000000000000000000";
     else
               out1 <= inc_bv (out1);
     end if;
end if;
end process;


process (trigg)                                                         ----------
begin
if (trigg'event and trigg = '1'  ) then
outA<= out1;
end if;
end process;

triggout<= debounce_trigg;

end contaclock_arch;
-------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;


ENTITY contapps is                                                      -------entity conta
pps
```

```
   port (
      pps: in STD_LOGIC;
      trigg: in STD_LOGIC;
      offset: in STD_LOGIC_VECTOR (31 downto 0);
      enable: in STD_LOGIC;
      reset: in STD_LOGIC;
      OUTpps: out STD_LOGIC_VECTOR (31 downto 0));
end contapps;

ARCHITECTURE contapps_arch of contapps is

signal out2:STD_LOGIC_VECTOR (31 downto 0);
signal enabledeb : std_logic;
signal offset1:STD_LOGIC_VECTOR (31 downto 0);

COMPONENT debounce
PORT (clock : IN std_logic;
     sign : IN std_logic;
     signout : OUT std_logic);
END COMPONENT;

function inc_bv (a: std_logic_vector) return std_logic_vector is
               variable s: std_logic_vector (31 downto 0);
               variable carry: std_logic;

     begin
               carry := '1';
               for i in 0 to 31 loop
                         s(i) := a(i) xor carry;
                         carry := a(i) and carry;
               end loop;
               return (s);
     end inc_bv;

function sum_bv ( a, b: std_logic_vector) return std_logic_vector is
               variable s: std_logic_vector (31 downto 0);
               variable carry : std_logic;

     begin
               carry := '0';
               for i in 0 to 31 loop
               s(i) := (a(i) xor b(31-i)) xor carry;
                         carry := ((a(i) or b(31-i)) and carry) or (a(i) and b(31-i));
               end loop;
               return (s);
     end sum_bv;


begin
A : debounce
     port map (pps, enable, enabledeb);

process(enable)
begin

if (enable'event and enable = '1') then
offset1 <= offset;
end if;
end process;

process(pps, reset, enable)
begin
```

```
if (reset = '1' or enable= '1') then
out2 <= "00000000000000000000000000000000";
elsif (pps'event and pps = '1' ) then
     if (out2 > "00000001000011110101111101000000" ) then
               out2 <= "00000000000000000000000000000000";
     else
               out2 <= inc_bv (out2);
     end if;
end if;
end process;

process (trigg)
begin
if (trigg'event and trigg = '1'  ) then
OUTpps<= sum_bv(out2,offset1);
end if;
end process;

end contapps_arch;



------------------------------------------------------------------- ENTITY PRINCIPALE
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

ENTITY generale is
     port (ppsIN,triggIN, clockIN,resetIN,selectIN,enableIN : in std_logic;
          outtrigg: out std_logic;
          offsetIN: in STD_LOGIC_VECTOR (31 downto 0);
          OUTconta: out STD_LOGIC_VECTOR (31 downto 0));
end generale;

ARCHITECTURE generale_arch of generale is
signal OUTclock:STD_LOGIC_VECTOR (31 downto 0);
signal OUTpps:STD_LOGIC_VECTOR (31 downto 0);

component contaclock
port (  pps: in STD_LOGIC;
        clock: in STD_LOGIC;
        trigg: in STD_LOGIC;
        triggout: out STD_LOGIC;
        reset: in STD_LOGIC;
        outA: out STD_LOGIC_VECTOR (31 downto 0));
end component;

component contapps
port(   pps: in STD_LOGIC;
     trigg: in STD_LOGIC;
     offset: in STD_LOGIC_VECTOR (31 downto 0);
     enable: in STD_LOGIC;
       reset: in STD_LOGIC;
       OUTpps: out STD_LOGIC_VECTOR (31 downto 0));
end component;

begin
C: contaclock
     port map (ppsIN,clockIN,triggIN,outtrigg,resetIN,OUTclock);
D: contapps
     port map (ppsIN,triggIN,offsetIN,enableIN,resetIN,OUTpps);
```

```
process (selectIN, OUTclock, OUTpps)
begin
if (selectIN = '0')  then
     OUTconta <= OUTclock;
else
     OUTconta <= OUTpps;
end if;
end process;

end generale_arch;
```

Di seguito è riportato il file con estensione “ucf” per l’assegnazione fisica dei piedini ai segnali di ingresso/uscita implementati nel listato del firmware:

```
NET clockIN LOC = P2;
NET ppsIN LOC = P39;
NET triggIN LOC= P70;
NET selectIN LOC=P106;
NET enableIN LOC=P76;

NET outtrigg LOC=P112;

NET offsetIN<0>   LOC = P4 ;          NET OUTconta<0>   LOC = P56 ;
NET offsetIN<1>   LOC = P5 ;          NET OUTconta<1>   LOC = P57 ;
NET offsetIN<2>   LOC = P9 ;          NET OUTconta<2>   LOC = P58 ;
NET offsetIN<3>   LOC = P10 ;         NET OUTconta<3>   LOC = P59 ;
NET offsetIN<4>   LOC = P12 ;         NET OUTconta<4>   LOC = P60 ;
NET offsetIN<5>   LOC = P13 ;         NET OUTconta<5>   LOC = P61 ;
NET offsetIN<6>   LOC = P14 ;         NET OUTconta<6>   LOC = P62 ;
NET offsetIN<7>   LOC = P15 ;         NET OUTconta<7>   LOC = P63 ;
NET offsetIN<8>   LOC = P16 ;         NET OUTconta<8>   LOC = P65 ;
NET offsetIN<9>   LOC = P19 ;         NET OUTconta<9>   LOC = P66 ;
NET offsetIN<10>   LOC = P20 ;        NET OUTconta<10>  LOC = P67 ;
NET offsetIN<11>   LOC = P21 ;        NET OUTconta<11>  LOC = P68 ;
NET offsetIN<12>   LOC = P22;         NET OUTconta<12>  LOC = P69;
NET offsetIN<13>   LOC = P23;         NET OUTconta<13>  LOC = P77 ;
NET offsetIN<14>   LOC = P24;         NET OUTconta<14>  LOC = P78;
NET offsetIN<15>   LOC = P25 ;        NET OUTconta<15>  LOC = P80 ;
NET offsetIN<16>   LOC = P26 ;        NET OUTconta<16>  LOC = P82 ;
NET offsetIN<17>   LOC = P28;         NET OUTconta<17>  LOC = P83;
NET offsetIN<18>   LOC = P29 ;        NET OUTconta<18>  LOC = P86 ;
NET offsetIN<19>   LOC = P30 ;        NET OUTconta<19>  LOC = P87 ;
NET offsetIN<20>   LOC = P31 ;        NET OUTconta<20>  LOC = P89 ;
NET offsetIN<21>   LOC = P32 ;        NET OUTconta<21>  LOC = P94 ;
NET offsetIN<22>   LOC = P41 ;        NET OUTconta<22>  LOC = P95 ;
NET offsetIN<23>   LOC = P42 ;        NET OUTconta<23>  LOC = P97 ;
NET offsetIN<24>   LOC = P43 ;        NET OUTconta<24>  LOC = P98 ;
NET offsetIN<25>   LOC = P46 ;        NET OUTconta<25>  LOC = P99 ;
NET offsetIN<26>   LOC = P47 ;        NET OUTconta<26>  LOC = P103 ;
NET offsetIN<27>   LOC = P48 ;        NET OUTconta<27>  LOC = P104 ;
NET offsetIN<28>   LOC = P49 ;        NET OUTconta<28>  LOC = P113 ;
NET offsetIN<29>   LOC = P50 ;        NET OUTconta<29>  LOC = P114 ;
NET offsetIN<30>   LOC = P51 ;        NET OUTconta<30>  LOC = P120 ;
NET offsetIN<31>   LOC = P52 ;        NET OUTconta<31>  LOC = P119 ;
```

BIBLIOGRAFIA

[1] Antonio Zichichi - progetto EEE (Extreme Energy Events) "La scienza nelle scuole"

[2] Jean Marie Zogg - "GPS Basic Intrduction to the system Application overiew"

[3] Micro Engineering Labs - "PicBasic Pro Compiler"

[4] www. Xilinx.com

[5] Douglas L. Perry - "VHDL: Programming by Example"

RINGRAZIAMENTI

Un primo e doveroso ringraziamento al

Prof. Marco Panareo

per la costante disponibilità e cortesia avute nei miei confronti e per aver svolto il compito di relatore con grande impegno.
Ringrazio inoltre tutto il gruppo di lavoro del laboratorio di elettronica dell'INFN per il supporto tecnico, in particolare il
Dott. Alessandro Corvaglia per il suo contributo fondamentale alla riuscita di questo lavoro, per avermi seguito e consigliato costantemente aiutandomi a superare tutte le difficoltà incontrate.

Grazie a tutti gli amici, compagni di studi e di vita che mi sono stati vicini in tutti i momenti di necessità e di felicità.

Un particolare ringraziamento alla mia famiglia, in particolare a mia madre Anna e mio padre Salvatore, che mi hanno sostenuto in tutti questi anni affrontando sacrifici senza i quali non avrei raggiunto questo traguardo.
Un ultimo ma non meno importante grazie a Manuela che mi è stata sempre accanto come nessun altro poteva fare.