UNIVERSITÀ DEGLI STUDI DI NAPOLI

FEDERICO II

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica Magistrale
Tesi di Laurea

Progettazione e Realizzazione di un Sistema di Verifica per il Radar Doppler Altimetro relativo alla missione Exomars (2016)

Relatore:
Ch.mo Prof. Davide De Caro

Correlatore:
Ing. Luca Ciofaniello

Candidato:

Antonio Scaldaferri
Matr. M61/000071

Anno Accademico 2011-2012

SOMMARIO

Introduzione

Il crescente interesse per l'esplorazione del Sistema solare e del pianeta Marte in particolare, hanno spinto potenze come la Russia, gli Stati Uniti, l'Europa, il Giappone a intraprendere missioni, organizzare spedizioni, individuare finestre di lancio di minima energia, progettare e inviare sonde automatiche senza equipaggio, orbiter, lander e rover per raccogliere dati.

E' in questo contesto che va inserito Aurora : un ambizioso programma di esplorazione spaziale dell'ESA in collaborazione con la NASA, basato su sonde automatiche, finalizzato all'esplorazione umana del Sistema solare e, in particolare, del pianeta Marte.

A tale scopo sono state pianificate tre principali missioni robotizzate per lo sviluppo di strategie di esplorazione robotica del Sistema Solare e del Pianeta Rosso:

- Exomars
- Mars Sample Return
- Missioni Arrow

La missione senza equipaggio Mars Sample Return (MSR), il cui lancio è previsto per il 2020-22, raccoglierà rocce e campioni del suolo marziano che, per la prima volta, verranno riportati sulla Terra e analizzati.

Le Missioni Arrow sono delle dimostrazioni tecniche concentrate sullo sviluppo di tecnologie necessarie per le missioni principali.

In particolare, la missione ExoMars, senza equipaggio, è finalizzata all'esplorazione della superficie marziana. Essa è basata su un rover, la cui caratteristica principale è la capacità di prendere decisioni autonome. Il lancio, inizialmente previsto per il 2011, è stato posticipato al 2016. Per motivi di natura prevalentemente economica, l'ESA e la NASA hanno stabilito di procedere sinergicamente nella futura esplorazione di Marte. I lanci sono previsti nel 2016, 2018 e 2020 e la NASA provvederà alla costruzione dell'orbiter e alla fornitura del razzo vettore.

Nello specifico, la missione ExoMars prevede sostanzialmente due fasi:

- nel 2016, l'invio di un orbiter e di un dimostratore tecnologico di ingresso e discesa, su Marte
- nel 2018, l'invio di due rover, uno europeo ed uno della NASA, su Marte.

Tale missione consentirà all'Europa di acquisire nuove tecnologie per l'esplorazione di Marte; in particolare il sistema di ingresso, discesa e atterraggio (EDLS) sarà testato nella prima missione, mentre il sistema Drill and Sample Preparation and Distribution (SPDS) sarà testato sul rover della seconda missione.

L'Italia partecipa al progetto con un finanziamento di 281 milioni di euro e ospiterà un centro di controllo.

Il CO.RI.S.T.A. (COnsorzio di RIcerca su Sistemi di Telesensori Avanzati) è coinvolto nelle attività di test del Radar Doppler Altimeter (RDA) di EXOMARS. Nel dettaglio, è responsabile della progettazione e sviluppo del "cuore" del sistema di test del Radar Doppler Altimeter (RDA) di EXOMARS, ovvero l'Echo Generator, che permette di simulare gli echi provenienti dalla superficie di Marte ed porli in ingresso al radar per testare lo strumento in varie condizioni operative.

Nel presente elaborato vengono illustrate le varie fasi che hanno consentito la realizzazione e il test dell'Echo Generator, con particolare riferimento:

- alla ricerca di un modello efficace per la simulazione degli echi riflessi dal suolo marziano
- allo studio dell'hardware disponibile per l'implementazione dell'Echo Generator,
- all'analisi delle tecniche di programmazione con l'utilizzo del modulo Labview FPGA, e alla loro applicazione per la creazione del programma
- alla valutazione, per mezzo di test, del funzionamento del sistema e delle performance che esso può raggiungere.

L'Echo Generator, infatti, permette di acquisizione e memorizzazione su disco degli impulsi generati dal radar e la simulazione degli echi provenienti dalla superficie di Marte. Tale sistema ricevere in ingresso un segnale analogico, campionato e acquisito, fornendo, poi, in uscita un segnale analogico di eco. L'hardware utilizzato per la realizzazione dell'Echo Generator, è basato su uno standard PXI. Con tale standard è possibile implementare due tipi di programma: un primo eseguito su FPGA e un secondo eseguito su processore (su cui è installato il sistema operativo Windows 7).

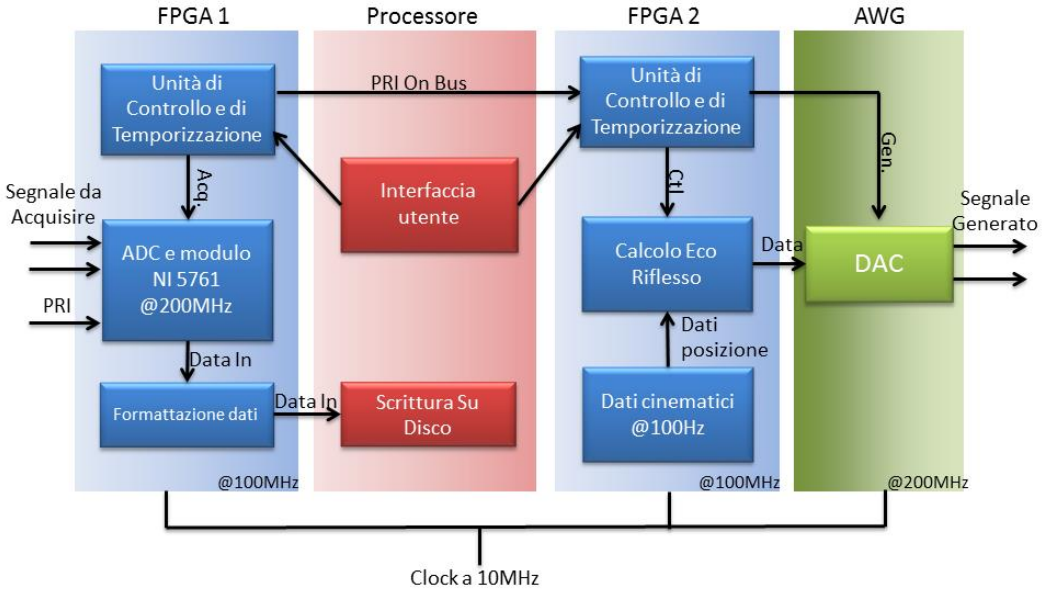

Figura I Schema a blocchi dell'Echo Generator sul sistema PXI

Nello schema a blocchi dell'Echo Generator (Figura I), si possono individuare le seguenti unità hardware:

- Il processore, su cui è implementata l'interfaccia utente e la scrittura su disco
- Le FPGA, utilizzate per eseguire le operazioni di controllo sull'acquisizione e sulla generazione.
- Arbitrary Waveform Generator per la generazione dei segnali di eco. Il trasferimento dei dati dall'FPGA all'AWG avviene a livello hardware, senza l'utilizzo del processore.

In particolare, la presentazione del lavoro di tesi è stato articolato in quattro fasi.

Inizialmente, vengono presentate le specifiche per la realizzazione dell'Echo Generator. Si fa, poi, riferimento alle modalità di integrazione dell'EG con il radar per i test e al modello teorico utilizzato per il calcolo degli echi riflessi dal suolo marziano. Vengono riportate le approssimazioni eseguite sul modello per semplificare l'implementazione su FPGA.

La seconda parte del lavoro concerne la descrizione del sistema hardware utilizzato, basato sulla piattaforma PXI e prodotto da National Instruments. L'architettura di tale sistema hardware è articolata in tre parti: lo chassis con corrispondente backplane per i collegamenti tra i vari moduli; il controller di sistema che può essere integrato o con controllo remoto da laptop o da PC; infine i moduli periferici utilizzati per integrare le FPGA, il convertitore Analogico-Digitale e l'AWG nel sistema. È stata inoltre descritta l'FPGA utilizzata, le diverse procedure di ottimizzazione realizzate e le risorse logiche, di cui l'FPGA dispone in dotazione. Viene, inoltre, presentato l'ambiente di sviluppo utilizzato per la programmazione delle risorse fisiche, messo a disposizione da NI. Per poter fruire delle risorse presenti sull'FPGA, è stato utilizzato il modulo Labview FPGA, espressamente creato a tale scopo. In questa fase, vengono anche mostrate le principali tecniche di programmazione di target FPGA con l'utilizzo del linguaggio grafico.

In una terza parte, viene descritto il progetto Labview necessario per implementare l'echo Generator nel sistema PXI. Sono stati sviluppati diversi programmi per configurare correttamente sia le FPGA, sia il processore sia l'AWG. In questa sezione, viene presentato il codice per la descrizione dei modelli per il calcolo degli echi. Sono mostrate, con ulteriore dettaglio, varie versioni/stesure dall'algoritmo, necessarie per giungere ad una versione finale, realmente implementabile.

Nella quarta ed ultima parte del lavoro, vengono descritti i test per la verifica del funzionamento dell'intero sistema, proposte le varie rappresentazioni degli echi generati attraverso FPGA, visualizzati i dati acquisiti con dei segnali di prova impiegati per testare la fase di acquisizione e memorizzazione su disco. Infine, viene proposta un'ultima prova sul sistema, chiuso ad anello, eseguita riportando in ingresso i segnali d'uscita dell'Echo Generator.

1 . Exomars

La missione

ExoMars è la prima missione del Programma Aurora Exploration, avviato all'Agenzia Spaziale Europea (ESA) in collaborazione con la NASA. La finalità primaria del Programma è di ideare e sviluppare un piano a lungo termine per la creazione di una serie di missioni robotiche, con alto sviluppo tecnologico e ricerche di grande interesse scientifico, che possano essere la base, o di supporto, per un'eventuale esplorazione umana nello spazio. L'Agenzia Spaziale Europea ha organizzato il programma ExoMars la cui finalità è lo studio preliminare dell'ambiente marziano con l'ausilio delle più avanzate tecnologie che in questa missione verranno testate in vista di una spedizione, da effettuarsi negli anni dal 2020 in poi, che possa consentire di portare sulla Terra reperti prelevati da Marte. Questo primo passo potrà permettere più approfondite sperimentazioni che potranno eventualmente predisporre all'invio di uomini verso il pianeta rosso. Questa missione prevede l'invio di un Orbiter ed un sistema di Entry, Descent and Landing Demonstrator Module (EDM) su Marte nel 2016 e due Rover, uno europeo ed uno della NASA nel 2018. L'Orbiter trasporterà strumenti scientifici per individuare e studiare tracce di gas atmosferico come, ad esempio, il metano, mentre l' Entry, Descent and Landing Demonstrator Module (EDM) trasporterà sensori in grado di valutare le prestazioni del lander durante la discesa e di studiare le caratteristiche ambientali del sito di atterraggio. Gli obiettivi scientifici principali della missione sono lo studio dell'ambiente biologico della superficie e la ricerca di eventuali tracce di vita, passata o presente; la caratterizzazione geochimica del pianeta e la distribuzione dell'acqua; la valutazione di possibili pericoli sulla superficie in previsione di future missioni con equipaggio; l'ampliamento della conoscenza dell'ambiente e della geofisica marziana. Inoltre la missione permetterà di sviluppare innovazioni tecnologiche per l'atterraggio di grandi carichi su Marte, lo sviluppo di energia solare sulla superficie e di migliorare la mobilità sul terreno marziano.

All'interno del sistema EDM è presente un radar doppler altimetro, che sarà il dispositivo per cui progettare il dispositivo di test, per questa trattazione. Il sistema EDM inizierà il suo viaggio nel gennaio del 2016, insieme al Trace Gas Orbiter. I dispositivi arriveranno approssimativamente nove mesi dopo su Marte e inizieranno la loro vera missione. Tre giorni prima che sia raggiunta l'atmosfera marziana, il modulo EDM sarà separato dall'Orbiter. La seconda fase, descritta come Entry, permetterà al modulo EDM di entrare nell'atmosfera marziana. La terza fase, detta Descent, sarà raggiunta dopo un'ulteriore decelerazione. In questa fase sarà attivato il Radar Doppler Altimetro, per permettere al modulo EDM di capire la sua posizione rispetto al suolo marziano e sarà utilizzato un paracadute per ridurre ulteriormente la velocità di caduta dell'EDM. L'ultima fase, il Landing, prevede l'avvio del sistema di propulsione a liquido per ridurre la velocità vicina allo zero e eseguire un atterraggio controllato. L'atterraggio avverrà in un sito ad alto interesse scientifico (un sito in cui è presente ematite che, sulla terra, si forma comunemente sotto bacini contenenti acqua). La Figura 1.1 mostra le varie fasi di discesa su Marte dell'EDM.

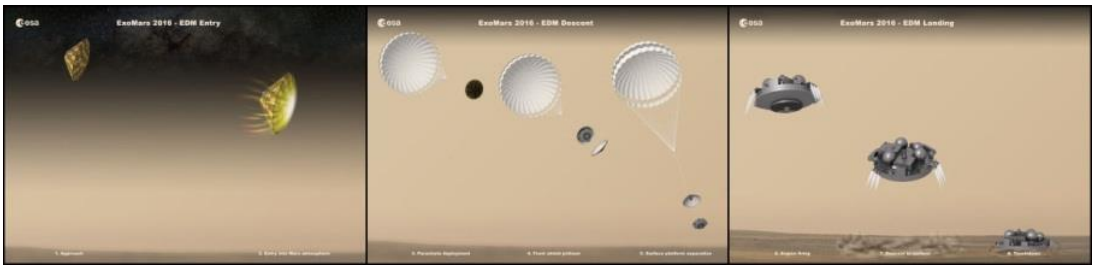

Figura 1.1 Le fasi di Entry, Descent e Landing del modulo EDM

Il Consorzio Corista

Il CO.RI.S.T.A. (COnsorzio di RIcerca su Sistemi di Telesensori Avanzati) è un consorzio senza fini di lucro. È' stato costituito a Napoli nel 1988, con lo scopo di promuovere una stretta cooperazione fra le Università e le Industrie nei settori legati al telerilevamento aerospaziale. Ha avuto, pertanto, fin dall'inizio, una rappresentanza di membri universitari e di membri industriali, che hanno contribuito, ognuno secondo le proprie peculiarità e attitudini, a fare del consorzio una struttura estremamente flessibile e ricca di risorse diversificate e specializzate.

Il consorzio si occupa della progettazione, lo sviluppo prototipale e l'applicazione di avanzati sensori per il telerilevamento che possono essere imbarcati su piattaforme terrestri, aeree e spaziali (palloni, aerei, elicotteri e satelliti). Svolge attività di ricerca e sviluppo prototipale nel settore dei sistemi di telerilevamento, comprendendo anche gli aspetti riguardanti il loro controllo in operazione e all'elaborazione dei dati da essi generati. Le aree di competenza tecnica, nell'ambito del telerilevamento, vanno dalla definizione e simulazione di sensori innovativi da aereo e di prototipi di sottosistemi da satellite allo sviluppo di algoritmi di elaborazione dati.

Il CO.RI.S.T.A. è coinvolto nelle attività di test del Radar Doppler Altimeter (RDA) di EXOMARS.
In particolare, ha il compito di progettare, sviluppare e testare un Echo Simulator Systems (ESS).

Il Sistema di Echo Simulator è composto dai seguenti sottosistemi:

- Radio Frequency Front End (RFFE);
- Signal Reference Distribution Unit (SDRU);
- Rack Power Distribution Unit (RPDU);
- Echo Generator (EG)

che servono in vario modo per permettere la trasmissione dei dati tra il radar e l'ESS. Il CO.RI.S.T.A. è responsabile della progettazione e sviluppo del "cuore" del sistema di test del Radar Doppler Altimeter (RDA) di EXOMARS, cioè l'Echo Generator, che permette di simulare gli echi provenienti dalla superficie di Marte e, iniettarli nel radar, per testare lo strumento in varie condizioni operative. Il lavoro di tesi è stato quello di progettare e realizzare l'Echo Generator e di testare lo stesso.

Radar Doppler Altimetro

Al fine di elaborare un sistema di test efficace, è utile comprendere il funzionamento di un Radar RDA altimetro.

L'altimetria si occupa di misurare la quota assoluta di un punto rispetto ad una superficie di riferimento. Quindi, Il principale obiettivo del radar altimetro è di misurare con grande precisione, attraverso l'invio di un impulso verso la superficie, il ritardo, la potenza e la forma d'onda dell'eco ricevuto. Attraverso questi parametri, che sono legati alla distanza dell'altimetro dalla superficie osservata, alla velocità del vento e ad altre variabili ambientali, si vuole calcolare i parametri d'interesse.

Nei radar altimetri convenzionali viene studiata la risposta del terreno ad un impulso, trasmesso da una antenna del radar, per valutare l'altezza. Come mostrato in Figura 1.2, inizialmente il segnale incidente illuminerà una superficie circolare.

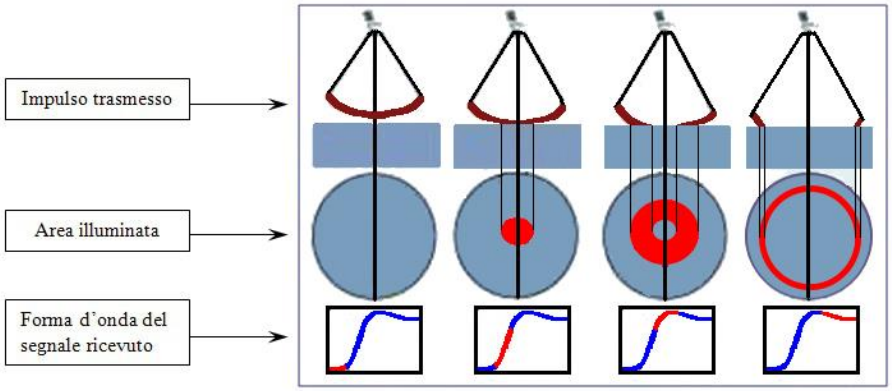

Figura 1.2 Rappresentazione teorica, di un radar altimentro convenzionale, dell'area illuminata e della forma d'onda del segnale ricevuto

In seguito, a mano a mano che il fronte dell'onda si sposta in avanti, la superficie illuminata diventerà una corona circolare che si allarga, fino al limite corrispondente all'apertura del fascio di antenna. L'energia riflessa da una superficie piatta, la cui rugosità è dello stesso ordine della lunghezza d'onda del radar, è proporzionale all'area illuminata. Quindi, l'ampiezza dell'eco di ritorno crescerà, inizialmente, quasi istantaneamente, e poi si manterrà costante, in quanto le aree delle superfici illuminate in istanti diversi sono uguali (aumenta il raggio delle corone circolari illuminate ma diminuisce lo spessore). In realtà l'andamento dell'eco non risulterà piatto a causa delle dimensioni finite dell'antenna. La posizione del centro del fronte di salita corrisponde alla distanza (media) dalla superficie di riferimento.

Il radar ad apertura sintetica (SAR) è un sensore di tipo attivo a microonde, e viene utilizzato nell'osservazione del suolo terrestre da un velivolo in movimento. Ha un'alta risoluzione azimutale, dovuta a una tecnica di elaborazione coerente dei segnali di ritorno dai singoli elementi del terreno osservati. Poiché il veicolo è in moto, i singoli elementi del terreno riflettono gli impulsi del radar più volte sotto diverse angolazioni, fornendo un'evoluzione doppler che va sotto il nome di "storia doppler". La frequenza doppler è la differenza tra la frequenza del segnale ricevuto dall'osservatore

e quello inviato dalla sorgente. Questa differenza è dovuta ad una variazione della distanza relativa nel tempo tra la sorgente e l'osservatore.

L'approccio dell'apertura sintetica viene applicato ai radar altimetri, utilizzando un'antenna puntata verso la direzione del nadir, ossia la direzione perpendicolare al terreno e passante per l'osservatore. L'apertura reale dell'antenna, nella direzione del moto del velivolo, viene suddivisa in più fasci, ognuno dei quali illumina un'area, la cui ampiezza dipende solo dalla modulazione Doppler, causata dal moto dell'altimetro. In un radar altimetro ad apertura sintetica, le corone circolari sono divise, dai fasci Doppler, in strisce perpendicolari alla direzione del moto. L'area delle superfici limitate dalle strisce e dagli archi di corona circolare diminuisce rapidamente all'allontanarsi lungo la direzione perpendicolare al moto. Di conseguenza l'eco di ritorno non presenterà l'andamento visto per i radar altimetri convenzionali, bensì avrà una rapida caduta come mostrato in Figura 1.3 e Figura 1.4.

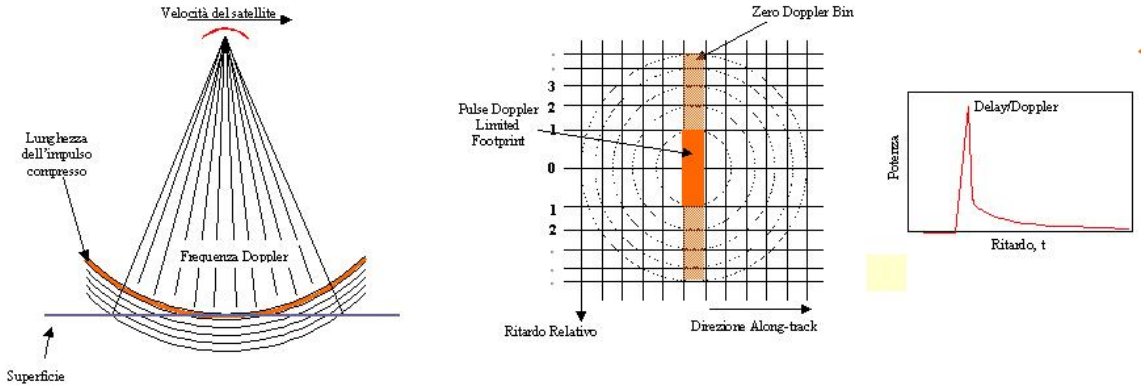

Figura 1.3 Rappresentazione di un Radar Doppler Altimetro RDA

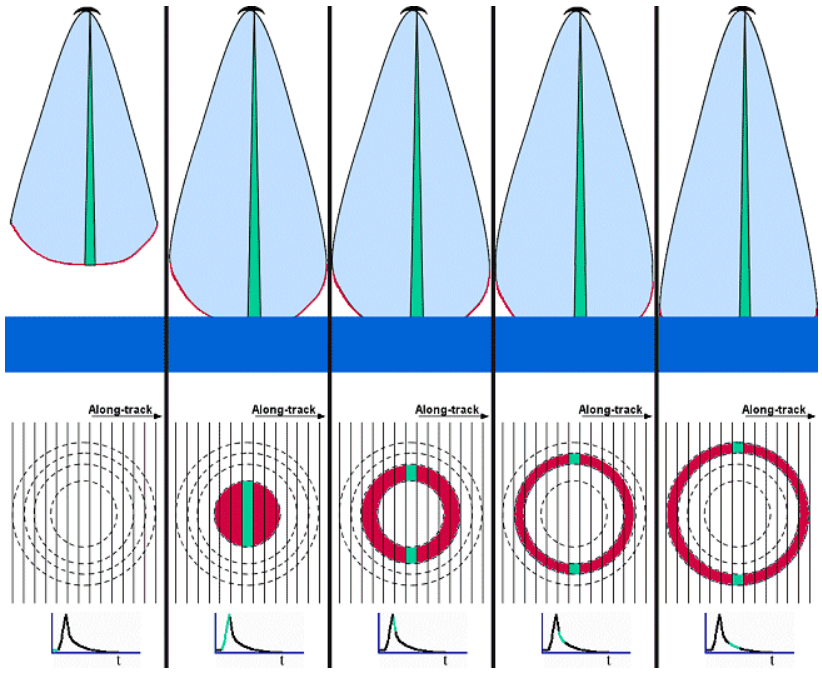

Figura 1.4 Funzionamento teorico, di un RDA, dell'area illuminata e della forma d'onda del segnale ricevuto

Il radar RDA altimetro, dispositivo per cui progettare il sistema di test, si basa sulla trasmissione di un impulso non modulato, e la ricezione dell'eco riflesso dal terreno. Le caratteristiche temporali dell'eco riflesso (ritardo temporale e forma), vengono utilizzate per calcolare la distanza di Slant Range, la distanza tra l'antenna e il terreno nella direzione dell'antenna. La correlazione complessa di echi successivi (generati da impulsi successivi) permette di ottenere la componente doppler. Delle relazioni matematiche permettono di passare da queste misurazioni, alla valutazione della quota del radar e ad un vettore velocità. L'RDA prevede un'unica catena di ricezione e di trasmissione connessa, attraverso una matrice di switch e circolatori, alle quattro antenne utilizzate dal radar. Una delle antenne è dedicata alla misura di distanza (per la funzione altimetro). Le rimanenti tre antenne sono utilizzate per eseguire una misura di velocità (per la funzione di radar doppler). L'antenna dedicata per la misura di distanze, indicata come Beam 0, è caratterizzata da parametri specifici temporali (PRF, tempo di persistenza) e della forma d'onda (larghezza dell'impulso) dell'impulso inviato. Le altre tre antenne, dedicate per la misura della velocità, sono utilizzate per monitorare dei parametri normalmente differenti da Beam 0. Una rappresentazione della geometria delle antenne è mostrata in Figura 1.5.

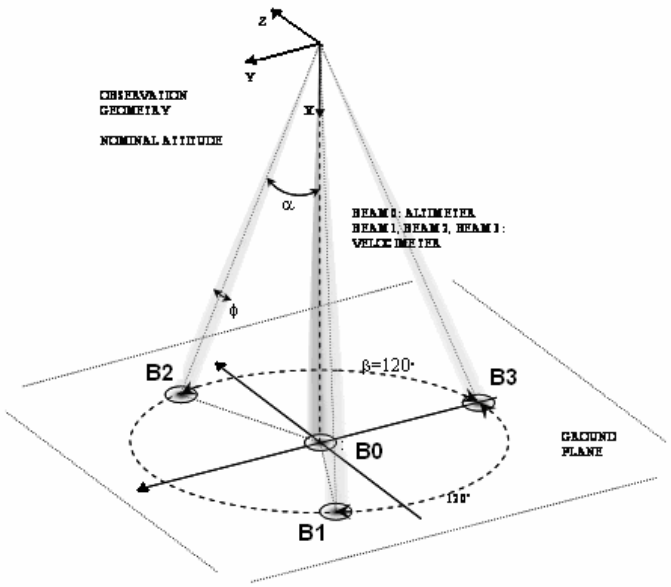

Figura 1.5 Strategia dell'osservazione della superficie

L'accesso ad ogni singola antenna si verifica, iterativamente, dopo un certo intervallo di tempo. In questo intervallo viene eseguita la trasmissione, la ricezione e l'elaborazione del segnale a/da ogni singola antenna.

I Parametri del radar RDA sono:

- Una portante a 35.76GHz (per massimizzare la potenza dell'eco riflesso e la risoluzione della misura della velocità)
- Quattro antenne (per ottenere 4 misure di velocità indipendenti e 4 misure accurate di slant range)
- Direttività dell'antenna >34 dB (per massimizzare la potenza dell'eco riflesso)

- Larghezza dell'impulso trasmesso di 40÷2560 ns (Limite massimo per massimizzare la potenza dell'eco riflesso e il limite minimo per assicurarci di eseguire delle misure a bassa quota)
- Una PRI (distanza tra due impulsi trasmessi) di 10÷240µs

Unità di test digitale del radar doppler altimetro

L'oggetto del lavoro di tesi è la progettazione di un sistema che permetta di testare il radar doppler altimetro. Per l'integrazione con il radar RDA, il sistema deve rispondere ad alcune specifiche. Le specifiche dell'ESS sono descritte in alcuni documenti rilasciati da ThalesAlenia Space. In questi documenti, vengono descritte le specifiche richieste per tutte le parti affidate al CO.RI.S.T.A., incluso l'EG, oggetto della trattazione. Inoltre, nei documenti, vengono descritte le interconnessioni con cui il sistema si interfaccerà con il radar RDA sotto test. Nei documenti, viene descritta la configurazione per l'ESS in closed loop con il radar RDA integrato nel modulo EDM di EXOMARS. In Figura 1.6 si mostra come l'ESS si interfaccerà con il radar RDA e la configurazione dell'intero sistema.

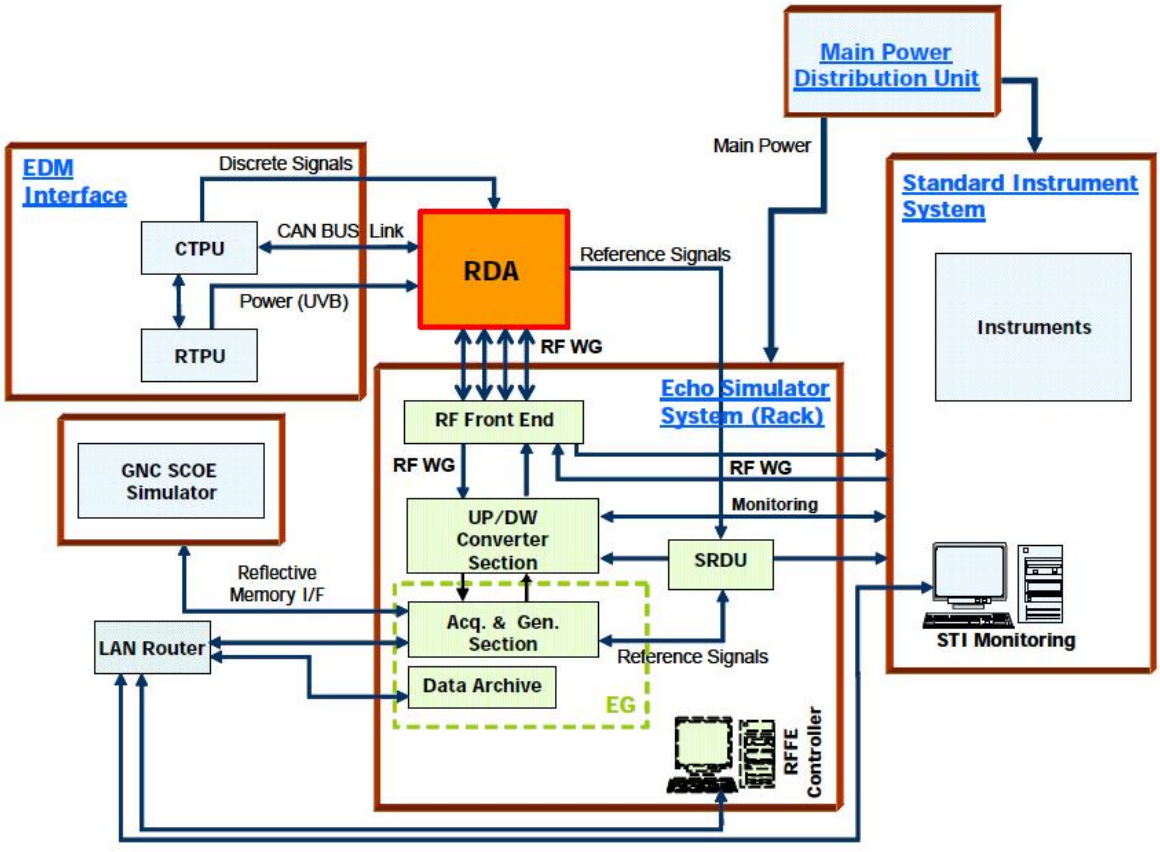

Figura 1.6 Integrazione dell'ESS con il Radar

Nella Figura 1.6 possiamo individuare quattro macro blocchi:

1. Main Power Distribution Unit (MPDU);
2. EDM Interface Simulator (EIS);

3. **Echo Simulator System (ESS);**
4. Standard Instrument System (STI).

L'unità MPDU fornirà l'alimentazione ad ognuno degli altri tre moduli.

Il simulatore dell'interfaccia con il modulo EDM (EIS) dovrà emulare le interfacce con il radar RDA. Inoltre dovrà archiviare le sequenze dei test eseguiti, i dati sensibili del test, in un database dedicato, e permettere di scegliere la sequenza dei test da eseguire. In particolare, l'RTPU fornirà l'alimentazione all'RDA e il CTPU fornirà i comandi e preleverà i dati di telemetria calcolati dal radar attraverso un collegamento CAN Bus dedicato. La scelta del test da eseguire verranno tradotte in informazioni sulla traiettoria (per esempio posizione e velocità iniziali) dall'EIS e, trasmesso all'EG, per il calcolo dell'eco da generare utilizzando un'interfaccia reflective memory I/F. La reflective memory è una memoria condivisa, basata su una struttura ad anello, con collegamenti in fibra ottica. Questa rete permette di condividere dei dati in real time, con una velocità di trasferimento fissata, indipendentemente dalla struttura del bus e dal sistema operativo utilizzato, quindi ottimale per le richieste del sistema di test.

L'unità di Echo Simulator System (ESS) è il cuore del sistema di test. È composto da un RF Front End per interfacciarsi con il radar (attraverso dei segnali a radiofrequenza). Quest'unità permette di interpretare i segnali di controllo, inviati dal radar, e di traslare gli impulsi inviati e gli echi ricevuti dalla banda di funzionamento del radar a 35.76 GHz, in banda base e viceversa, per eseguire l'elaborazione digitale dei segnali. L'EG genera l'eco riflesso del suolo marziano (simulato) come risposta al segnale dal radar, acquisisce e salva il segnale trasmesso dal radar (portato in banda base dall'RF Front End). Inoltre dovrà monitorare le operazioni di generazione e acquisizione dell'eco. L'RSDU, il Signal Reference Distribution Unit, permetterà di adattare i segnali di sincronizzazione ricevuti dal radar RDA (per esempio 50Mhz per sincronizzare il sistema, PRI Trigger, tipo d'impulso ed altri) nel tipo (LVTTL, single ended o differenziali) per redistribuirli all'EG e allo Standard Instrument System (STI).

Il Standard Instrument System (STI) si interfaccerà con l'RSDU, permetterà di monitorare l'avanzamento del test e valutare i risultati del test attraverso degli strumenti adatti.

Modello per lo studio della forma d'onda riflessa

Una volta chiarite le finalità del sistema da realizzare, risulta necessario individuare un modello preciso e efficace, da implementare su una piattaforma digitale, per simulare l'eco riflesso. Questa elaborazione dovrà essere eseguita in real time sull'EG. Il modello scelto, e poi utilizzato, è quello applicato per il radar altimetro utilizzato anche nella missione Cassini su Titano, satellite di Saturno. Nel caso del missione Cassini, questo algoritmo è stato utilizzato per interpretare correttamente i dati ottenuti con la missione. Come è noto, le caratteristiche di un'onda riflessa da una superficie sono strettamente legate alle proprietà statistiche della superficie, come asperità, valore medio della pendenza ed altri. In principio, questo significa che le informazioni contenute nella portante di un impulso ricevuto possono essere estratte, se viene utilizzato un modello dell'eco riflesso adeguato.

È necessario utilizzare un'analisi alternativa a quella classica perché alcune ipotesi, fatte nel modello classico, non possono essere applicate nel caso esaminato. Il radar RDA utilizzato nell'EDM, come nella missione Cassini, lavora in condizioni di "Beam-limited" e di "Pulsewidth-limited" comparabili, invece nel modello classico si considera una condizione di "Pulsewidth-limited". Questa condizione di lavoro porta il modello a risultati non realistici. Il modello, che verrà utilizzato, è estratto dall'articolo "A Waveform Model for Near-Nadir Radar Altimetry Applied to the Cassini Mission to Titan" di Giovanni Alberti, Luca Festa, Claudio Papa, and Guido Vingione pubblicato sul IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, VOL. 47, NO. 7 del giugno 2009. In questo articolo viene proposto un modello analitico alternativo per un'approssimazione di un eco riflesso da una superficie, valido in condizioni di "Beam-limited" e di "Pulsewidth-limited" comparabili.

Questo studio parte dalle osservazioni di G.S. Brown che, nel 1977, propose un modello teorico per lo studio di un'onda riflessa da una superficie irregolare. Questo modello fu subito chiamato il modello di Brown e fu applicato nel caso delle osservazioni oceaniche, con i radar altimetri che lavorano in condizione di "Pulsewidth-limited". Una direzione, utilizzata per lo studio degli echi riflessi, è quella del nadir, ossia la direzione perpendicolare al piano, da cui misurare la distanza, e passante per l'osservatore. L'approssimazione di una risposta ad una superficie piana , ottenuta con il modello e con il metodo di Laplace, in direzioni diverse dal nadir, è affetta da un errore minore del 2% rispetto al valore reale dell'eco riflesso.

Le ipotesi presenti nel modello di Brown sono:

- Meccanismi di scattering completamente incoerenti;
- Elementi indipendenti di scattering sulla superficie osservata;
- Irregolarità della superficie, descritta con una densità di probabilità gaussiana;
- Variazioni di frequenza per effetto doppler trascurabili.

Il valore medio di un eco riflesso, sia in direzione del nadir sia di off-nadir, può essere calcolato, nel caso generale, attraverso una convoluzione fra tre elementi: l'eco riflesso di una superficie piana (FSIR); la risposta della superficie, cui è rivolto il radar; la densità di probabilità dell'altezza dei punti speculari della superficie osservata.
L'espressione dell'FSIR, includendo errori di puntamento ξ con la direzione del nadir, è estratta dal modello di Brown e modifica in piccola parte. L'FSIR risulta una funzione della variabile τ. Le espressioni sono mostrate nell'Equazione 1.1.

$$\begin{cases} \mathrm{FSIR}(\tau) = K_{\mathrm{FS}} \exp\left(-\frac{4c\tau}{\gamma h\Lambda}\cos 2\xi\right) \cdot I_0\left(\frac{4}{\gamma}\sin 2\xi\sqrt{\frac{c\tau}{h\Lambda}}\right), & \tau \geq 0 \\ \mathrm{FSIR}(\tau) = 0, & \tau < 0 \end{cases}$$

$$\tau = t - 2h/c$$

Equazione 1.1

Dove

$$K_{\mathrm{FS}} = \frac{G_0^2 \lambda^2 c \sigma^0(\psi_0)}{4(4\pi)^2 L_p h^3} \exp\left(-\frac{4}{\gamma}\sin^2 \xi\right)$$

$$\psi_0 \approx \tan^{-1}(\sqrt{c\tau/h}).$$

$$\Lambda = (1 + h/R_T)$$

$$\gamma = -\frac{2\sin^2(\theta_{3\,\mathrm{dB}}/2)}{\ln(0.5)}$$

Equazione 1.2

In questo caso:

- c rappresenta la velocità della luce;
- λ rappresenta la lunghezza d'onda della portante del radar;
- L_p rappresenta le perdite del segnale, nel percorso radar-terreno-radar;
- h rappresenta l'altezza del satellite, sul livello medio della superficie osservata;
- σ_0 è la costante che tiene in conto dell'area effettivamente illuminata dal radar (dipende solo dall'angolo di osservazione ψ_0 che può essere trascurato in caso di piccoli angoli);
- G_0 rappresenta il picco del guadagno dell'antenna (sull'asse di simmetria dell'antenna);
- θ_{3dB} rappresenta l'apertura a -3dB dell'antenna.

L'unica differenza, con la classica formulazione classica di Brown, è l'inclusione degli effetti di una superficie di riflessione sferica, invece che piana, che comporta una sostituzione formale tra τ e τ/Λ con Λ definito nella terza dell'Equazione 1.2. In questo caso, R_T rappresenta il raggio medio di Marte.
È possibile ottenere l'eco riflesso (IR) come la convoluzione tra le funzioni FSIR e P_{HI}. La funzione P_{HI} rappresenta la convoluzione tra la risposta della superficie, cui è rivolto il radar, e la densità di probabilità dell'altezza dei punti osservati. Supponiamo queste due finzioni Gaussiane. L'espressione di P_{HI} è mostrata nell'Equazione 1.3.

$$P_{\mathrm{HI}}(\tau) = K_{\mathrm{HI}} \exp\left[-\frac{(\tau - \tau_0)^2}{2\sigma_c^2}\right]$$

Equazione 1.3

dove

$$K_{\mathrm{HI}} = P_T B T \sqrt{2\pi}\frac{\sigma_p}{\sigma_c}$$

$$\sigma_c = \sqrt{\frac{4}{c^2}\sigma_h^2 + \frac{1}{8\ln 2 B^2}}$$

Equazione 1.4

In questo caso:

- P_T rappresenta il picco di potenza trasmesso;
- B rappresenta la banda dell'impulso trasmesso;
- T rappresenta la larghezza temporale dell'impulso trasmesso.

Il parametro σ_c è legato al valore quadratico medio dell'altezza rispetto alla quota della superficie di riferimento σ_h, e alla risoluzione verticale del sistema (1/2B). Nell'Equazione 1.3, è presente il termine τ_0 che rappresenta un ritardo aggiuntivo a τ (tempo che intercorre tra l'invio e la ricezione dell'impulso dal radar nel caso in cui ξ=0). Il termine τ_0 è presente solo nel caso in cui ci sia un errore di puntamento ξ, ossia una differenza tra la direzione in cui viene inviato l'impulso e la direzione del nadir. L'angolo ξ è mostrato in Figura 1.7. Nel caso in cui non c'è un errore di puntamento (ξ=0) si assume τ_0 =0.

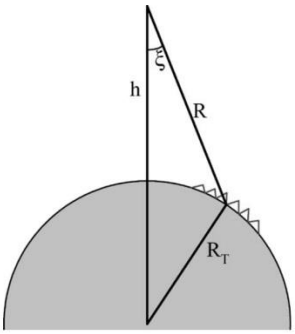

Figura 1.7 Geometria dell'altimetro se non risulta orientato in direzione del nadir

La convoluzione non può essere risolta analiticamente nel caso generale, ad eccezione della configurazione in cui il radar è diretto verso la direzione del nadir (ξ=0). In questo caso si ottiene l'Equazione 1.5 per l'eco riflesso.

$$IR^{\mathrm{Nadir}}(\tau) = P_{\mathrm{FS}}(\tau)\,|_{\xi=0} * P_{\mathrm{HI}}(\tau)$$
$$IR^{\mathrm{Nadir}}(\tau) = K\sigma^0 \frac{1}{2}\exp\left(-\frac{\delta}{\sigma_c}\tau + \frac{\delta^2}{2}\right)\cdot\left[1+\mathrm{erf}\left(\frac{\tau}{\sqrt{2}\sigma_c} - \frac{\delta}{\sqrt{2}}\right)\right]$$
$$K = \frac{G_0^2\lambda^2 c P_T T}{32\pi\sqrt{8\ln 2}L_p h^3}$$
$$\delta = \frac{4c}{\gamma h \Lambda}\sigma_c.$$

Equazione 1.5

Queste equazioni possono essere considerate come una generalizzazione del modello di Brown. L'Equazione 1.5 corrisponde esattamente a quelle ricavate dal modello di Brown, se si assume per vera la condizione di "Pulsewidth-limited", mostrata nell'Equazione 1.6, per il radar.

$$\frac{4c}{\theta^2_{3\,\mathrm{dB}} B h} \ll 1$$

Equazione 1.6

Nel caso in cui il radar non è diretto verso la direzione del nadir ($\xi\neq$0), l'espressione dell'FSIR non può essere semplificata. In questo caso, con il metodo di Laplace, è possibile calcolare una forma chiusa per l'FSIR. L'espressione è mostrata nell'Equazione 1.7.

$$\mathrm{FSIR}^{\mathrm{Asymp}}(\tau) = \frac{G_0^2 \lambda^2 c \sigma^0(\psi_0)}{2(4\pi)^3 L_p h^3} G(\varepsilon), \quad \tau \geq 0$$

Equazione 1.7

Dove

$$G(\varepsilon) = \exp\left(-\frac{4\left(\sin\xi - \varepsilon\cos\xi\right)^2}{\gamma(1+\varepsilon^2)}\right)\sqrt{\frac{2\pi}{a+2b}}$$
$$a = \frac{4\varepsilon}{\gamma}\frac{\sin 2\xi}{(1+\varepsilon^2)} \quad b = \frac{4\varepsilon^2}{\gamma}\frac{\sin^2\xi}{(1+\varepsilon^2)} \quad \varepsilon = \sqrt{\frac{c\tau}{h\Lambda}}.$$

Equazione 1.8

Per alti valori dell'angolo di off-nadir, il valore dell'FSIR diventa molto più grande della distribuzione dell'altezza e della risposta dell'impulso, che possono essere trascurati nella convoluzione. L'IR, considerando alti valori dell'angolo di off-nadir, si può esprimere come l'Equazione 1.9.

$$IR^{\mathrm{Asym}}(\tau) = \frac{K\sigma^0 G(\varepsilon)}{2}\left[1 + \mathrm{erf}\left(\frac{\tau}{\sqrt{2}\sigma_c}\right)\right] \qquad \tau \geq 0$$

Equazione 1.9

Il problema, di trovare una forma chiusa per valutare l'eco riflesso, rimane per piccoli angoli rispetto alla direzione del nadir. In questo caso, già dal testo di Brown, viene suggerita un'approssimazione dell'espressione dell'FSIR con una serie di esponenziali utilizzando il classico modello di Prony. In questo caso, l'espressione dell'FSIR dipende da un parametro, ε, adimensionale. L'FSIR può essere semplificata, nell'Equazione 1.10, come una somma di N termini esponenziali.

$$\mathrm{FSIR}^{\mathrm{Prony}}(\varepsilon) = K_{\mathrm{FS}} \exp(-K_a\tau) \sum_{i=1}^{N} C_i \exp(K_i\tau)$$

Equazione 1.10

Dove

$$\begin{cases} K_a = \frac{4}{\gamma}\cos(2\xi)\frac{c}{h\Lambda} \\ K_i = \frac{4}{\gamma}\sin(2\xi)\frac{c}{h\Lambda}a_i. \end{cases}$$

Equazione 1.11

I parametri a_i e C_i sono dei fattori calcolati con il metodo di Prony e sono caratteristici per la quota, per l'angolo di off-nadir e per l'ordine dell'approssimazione scelto. I parametri K_i possono essere sia reali sia immaginari, ma la somma degli N termini deve dare necessariamente un risultato reale.

L’integrale di convoluzione può essere valutato, con queste premesse, come nel caso in cui l’angolo di off-nadir è pari a zero. L’espressione di IR può essere scritta come l’Equazione 1.12.

$$IR^{\text{Prony}}(\tau)=\frac{K\sigma^0}{2}\exp\left(-\frac{4}{\gamma}\sin^2\xi\right)\cdot\sum_{i=1}^{N}C_i\exp\left(-\frac{\delta_i}{\sigma_c}\tau+\frac{\delta_i^2}{2}\right)\left[1+\text{erf}\left(\frac{\tau}{\sqrt{2}\sigma_c}-\frac{\delta_i}{\sqrt{2}}\right)\right].$$

Equazione 1.12

Dove

$$\delta_i=(K_a-K_i)\sigma_c$$

Equazione 1.13

Nel caso della nostra analisi, per semplificare l’implementazione, viene utilizzata un’unica espressione per IR, quella per grandi angoli di off-nadir. La scelta di questo modello è supportata da alcune considerazioni generali sul sistema. Il radar, utilizzando quattro antenne e unica catena di generazione del segnale, ha solo un quarto delle probabilità di avere un’antenna diretta nella direzione del nadir. Le altre antenne, essendo orientate in direzioni differenti, avranno un valore dell’angolo di off-nadir grande. Attraverso queste considerazioni, il modello approssimato per angoli grandi è il più adatto.

Approssimazioni Eseguite sul modello

Il modello scelto, per calcolare l'eco riflesso dal terreno, è quello approssimato per grandi angoli di off-nadir. Le espressioni, da cui verranno desunte quelle inserite nel dispositivo di test dell’RDA, vengono presentate nuovamente nelle Equazione 1.14, Equazione 1.15, Equazione 1.16, Equazione 1.17, Equazione 1.18, Equazione 1.19 e Equazione 1.20 per maggiore chiarezza.

$$IR^{\text{Asym}}(\tau)=\frac{K\sigma^0G(\varepsilon)}{2}\left[1+\text{erf}\left(\frac{\tau}{\sqrt{2}\sigma_c}\right)\right]\qquad\tau\geq0$$

Equazione 1.14

$$\tau=t-2h/c$$

Equazione 1.15

$$G(\varepsilon)=\exp\left(-\frac{4\left(\sin\xi-\varepsilon\cos\xi\right)^2}{\gamma(1+\varepsilon^2)}\right)\sqrt{\frac{2\pi}{a+2b}}$$

$$a=\frac{4\varepsilon}{\gamma}\frac{\sin2\xi}{(1+\varepsilon^2)}\quad b=\frac{4\varepsilon^2}{\gamma}\frac{\sin^2\xi}{(1+\varepsilon^2)}\quad\varepsilon=\sqrt{\frac{c\tau}{h\Lambda}}.$$

Equazione 1.16

$$K=\frac{G_0^2\lambda^2cP_TT}{32\pi\sqrt{8\ln2}L_ph^3}$$

Equazione 1.17

$$\Lambda = (1 + h/R_T)$$

Equazione 1.18

$$\sigma_c = \sqrt{\frac{4}{c^2}\sigma_h^2 + \frac{1}{8 \ln 2 B^2}}$$

Equazione 1.19

$$\gamma = -\frac{2 \sin^2(\theta_{3\,\mathrm{dB}}/2)}{\ln(0.5)}$$

Equazione 1.20

L'algoritmo per il calcolo dell'eco riflesso verrà eseguito periodicamente sull'EG. È importante capire, quindi, come ottimizzarlo e, per questo, conoscere gli input dell'algoritmo e i loro aggiornamenti, che dipenderanno dalla posizione del radar e dalle caratteristiche dello stesso. Nelle Equazione 1.14, Equazione 1.15, Equazione 1.16, il simbolo τ indica il vettore che tiene in conto degli istanti di tempo. Il valore di τ, perciò, cambierà durante il calcolo di ogni campione degli echi riflessi. Il valore "h", che rappresenta la distanza tra il radar e la superficie di riferimento sulla direzione ortogonale alla stessa, rimane costante durante il calcolo dei campioni di ogni eco riflesso, ma cambia per impulsi successivi perché varia la posizione del radar. In questo modo, si possono considerare costanti, fissata la posizione del radar, K, Λ, σ_c e γ nel calcolo di ogni eco riflesso. Invece ε, e di conseguenza a e b, risultano dei vettori che dipenderanno dall'istante di tempo considerato τ, e varieranno per ogni campione calcolato.

L'approccio utilizzato per implementare efficacemente il calcolo è di semplificare l'elaborazione, dividendola in due parti. Una prima parte che calcola solo la forma dell'eco riflesso, e una seconda che permetta di calcolare la potenza reale del segnale. Il modello dell'eco riflesso viene utilizzato solo per estrarre la sua forma. Una volta calcolato l'eco riflesso, viene normalizzato e, in questo modo, si trascurano alcuni fattori semplificandone il calcolo. Attraverso la potenza reale del segnale e un prodotto, si calcolano i campioni reali dell'eco riflesso. In questo modo si semplifica il calcolo dei campioni, eseguito per ogni impulso, che possono essere elaborati più velocemente, senza però perdere informazioni.

Calcolo delle costanti

La forma dell'eco riflesso è il calcolo che richiederà maggiori risorse implementative. Dovendo valutare l'eco riflesso normalizzato, possono essere trascurate tutte le variabili, che rimangono costanti, durante il calcolo di ogni eco. Con queste considerazioni, possiamo trascurare le costanti K e σ^0 nell'elaborazione dell'Equazione 1.14.

Il termine Λ, nell'Equazione 1.18, può essere approssimato ad 1 perché il rapporto h/Rt, essendo h molto minore del raggio marziano, risulta molto minore di uno. Il raggio medio (tra raggio polare ed equatoriale) di Marte misura 3392,8km. e h varia tra 8500 e 200m. In conseguenza Λ è circa 1 e lo possiamo trascurare nel calcolo del parametro ε.

Il fattore γ è, invece, indipendente sia dal vettore tempo sia dalla quota dell'altimetro, ma dipende solamente dalle caratteristiche dell'antenna del radar. In particolare dipende dal parametro θ_{3dB}, ossia l'apertura a -3dB dell'antenna. Fissata l'antenna, il fattore γ risulta costante. Nel nostro caso θ_{3dB} è pari a 0,07433rad e il fattore γ risulta uguale a 0,003984. Nelle espressioni di G(ε),di a e di b, il fattore γ compare sempre come $4/\gamma$. Il valore di $4/\gamma$ è 1004,03.

Il vettore tempo τ

Il vettore τ e il vettore tempo t rappresentano il tempo, in cui il campione dell'eco riflesso verrà ricevuto dal radar. Questi vettori devono essere definiti, in modo tale, da considerare solo la parte utile del segnale, quella contenente l'eco riflesso, e trascurare gli istanti in cui il modello determinerebbe campioni nulli. I campioni nulli sono presenti prima dell'eco riflesso, nel caso in cui il segnale inviato non è ancora ritornato al radar; o dopo che il radar ha ricevuto tutto l'impulso inviato. Il primo campione non nullo del segnale calcolato è nell'istante t_{ric} (Equazione 1.21), il tempo necessario al segnare per essere riflesso dal terreno e ritornare al radar. Nell'Equazione 1.21, Z rappresenta la misura precedente del radar, ricevuta dall'antenna altimetro. Nel caso delle simulazioni, la Z viene comunicata all'unità di test attraverso l'interfaccia reflective memory. Sempre nell'Equazione 1.21, ξ rappresenta l'errore di puntamento dell'antenna altimetro e viene utilizzato per calcolare l'altezza effettiva, H, dell'altimetro.

$$t_{ric} = 2\frac{H}{c} = 2\frac{Z}{\mathrm{c} * \cos\xi}$$

Equazione 1.21

La lunghezza del vettore t deve essere tale da comprendere tutto l'inviluppo dell'eco. Quindi, la lunghezza di questo vettore si deve definire accuratamente affinché non sia né troppo piccola (per calcolare tutto l'eco riflesso e non solo una parte), né troppo grande (per evitare problemi computazionali). Il vettore tempo t viene definito come segue:

$$\mathrm{t} = \left[2\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi};\ 2\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi} + \mathrm{t_{ck}}; \dots\dots\dots\dots;\ 4\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi}\right]$$

Equazione 1.22

Dove t_{ck} rappresenta la distanza temporale tra due campioni calcolati. Nel caso di nostro interesse, dovendo generare i segnali a una frequenza di 200Mhz, il t_{ck} è pari a 5ns. Il primo istante di tempo considerato è proprio t_{ric}, l'ultimo istante viene imposto pari a 2* t_{ric} e, di conseguenza, si impone la dimensione massima dell'impulso inviato pari a t_{ric}. Il vettore τ, definito nell'Equazione 1.15, può essere espresso come:

$$\tau = \left[2\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi} - 2\frac{\mathrm{Z}}{\mathrm{c}};\ 2\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi} + \mathrm{t_{ck}} - 2\frac{\mathrm{Z}}{\mathrm{c}}; \dots\dots\dots\dots;\ 4\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi} - 2\frac{\mathrm{Z}}{\mathrm{c}}\right]$$

Equazione 1.23

Attraverso queste semplificazioni, possiamo riscrivere il modello come segue:

$$IR^{Asym}(\tau) = G(\varepsilon)\left[1 + \operatorname{erf}\left(\frac{\tau}{\sqrt{2}\sigma_c}\right)\right] \qquad \tau \geq 0$$

Equazione 1.24

$$G(\varepsilon) = \exp\left[-1004{,}03\left(\frac{(\sin\xi - \varepsilon\cos\xi)^2}{1+\varepsilon^2}\right)\right] \cdot \sqrt{\frac{2\pi}{a+2b}}$$

Equazione 1.25

$$a = 1004{,}03 \cdot \frac{\varepsilon \sin 2\xi}{1+\varepsilon^2}$$

Equazione 1.26

$$b = 1004{,}03 \cdot \frac{\varepsilon^2 \sin^2 \xi}{1+\varepsilon^2}$$

Equazione 1.27

$$\varepsilon = \sqrt{\frac{c \cdot \tau}{Z}}$$

Equazione 1.28

Valutazione dell'incidenza della funzione Erf(x) sul modello

Il successivo passo, per semplificare l'implementazione, è quello di capire se è possibile trascurare la funzione erf(x) nell'Equazione 1.24, per il calcolo dell'eco riflesso. Lo studio eseguito è quello di confrontare i valori rilevati nelle due versioni della IR^{Asym}, quella semplificata e quella non semplificata. Questa operazione è stata eseguita in Matlab, per avere una maggiore precisione. La funzione è stata implementata attraverso una sotto funzione, mostrata nella Figura 1.8.

```
function [Pr SWST] = Echo_FPGA(H,zita,R_ref,T,B,sigma_h,theta_3dB,fs,
SR = H / cos(zita);
gamma = 2/log(2)*sin(theta_3dB/2)^2;
t0 = 2*SR/c;
t = (t0*1+1e-20 : 1/fs : 4 * t0);
tau = t - t0;

eps = sqrt(c*tau/SR);
eps(tau<0) = 0;
a = 4*eps*sin(2*zita)/gamma./(1+eps.^2);
b = 4*eps.^2*sin(zita)^2/gamma./(1+eps.^2);
sigma_p = sqrt(1/log(2)/8)/B;
sigma_s = sigma_h*2/c;
sigma_c = sqrt(sigma_p^2+sigma_s^2);

if zita == 0
    alfa    = 4*c/gamma/H/(1+H/R_ref);
    delta   = alfa*sigma_c;
Pr = exp(-delta/sigma_c*tau + delta^2/2) .* ( 1 + erf(tau/sqrt(2)/sig
else
G_eps = exp( -4 * (sin(zita)- eps * cos(zita)).^2 / gamma ./ (1+eps.^
    G_eps(tau<0) = 0;

    %IR_Asym = G_eps .* (1 + erf(tau/sqrt(2)/sigma_c));
    IR_Asym = G_eps;

    Pr = IR_Asym;

end
SWST = 2*SR/c;
Pr  = Pr/max(Pr);
end
```

```
function [Pr_erf SWST_erf] = Echo_FPGA_ERF(H,zita,R_ref,T,B,sigma_h,t
SR = H / cos(zita);
gamma = 2/log(2)*sin(theta_3dB/2)^2;
t0 = 2*SR/c;
t = (t0*1+1e-20 : 1/fs : 4 * t0);
tau = t - t0;

eps = sqrt(c*tau/SR);
eps(tau<0) = 0;
a = 4*eps*sin(2*zita)/gamma./(1+eps.^2);
b = 4*eps.^2*sin(zita)^2/gamma./(1+eps.^2);
sigma_p = sqrt(1/log(2)/8)/B;
sigma_s = sigma_h*2/c;
sigma_c = sqrt(sigma_p^2+sigma_s^2);

if zita == 0
    alfa    = 4*c/gamma/H/(1+H/R_ref);
    delta   = alfa*sigma_c;
    Pr_erf = exp(-delta/sigma_c*tau + delta^2/2) .* ( 1 + erf(tau/sqr
else
    G_eps = exp( -4 * (sin(zita)- eps * cos(zita)).^2 / gamma ./ (1+e
    G_eps(tau<0) = 0;

    IR_Asym = G_eps .* (1 + erf(tau/sqrt(2)/sigma_c));
    %IR_Asym = G_eps;

    Pr_erf = IR_Asym;

end
SWST_erf = 2*SR/c;
Pr_erf  = Pr_erf/max(Pr_erf);
end
```

Echo_FPGA.m* | Echo_FPGA_ERF.m

Figura 1.8 Sottofunzioni Matlab per il calcolo l'eco riflesso nella versione semplificata e nella versione completa

La funzione a sinistra, nella Figura 1.8, permette di calcolare l'eco nella versione semplificata, quella a destra la versione completa, in cui è presente la funzione erf(x). Sono state valutate varie condizioni, cambiando sia lo slant range sia l'angolo di off-nadir, per trovare la condizione peggiore. Il parametro utilizzato, per valutare il peso della funzione erf(x), è la variazione percentuale. Vengono riportati i grafici degli impulsi calcolati dalle due versioni, sia in scala lineare, nella Figura 1.9, sia in scala logaritmica, nella Figura 1.10. Nella Figura 1.9 e Figura 1.10, viene mostrato il caso in cui si ha uno slant range pari a 4000m (lo slant range varia tra 200m e 8000m) e angolo di off-nadir di 40° (varia 0,05° e 55°).

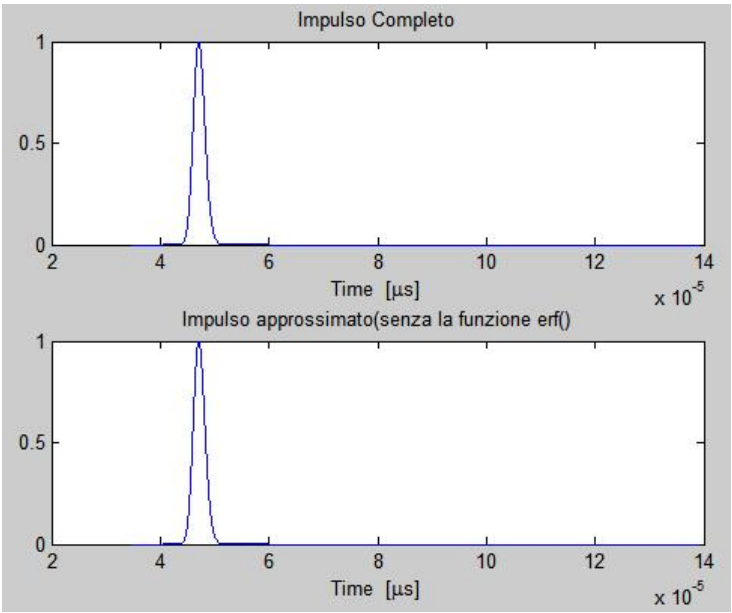

Figura 1.9 Rappresentazione dell'eco riflesso calcolato sia nella versione semplificata che nella versione completa

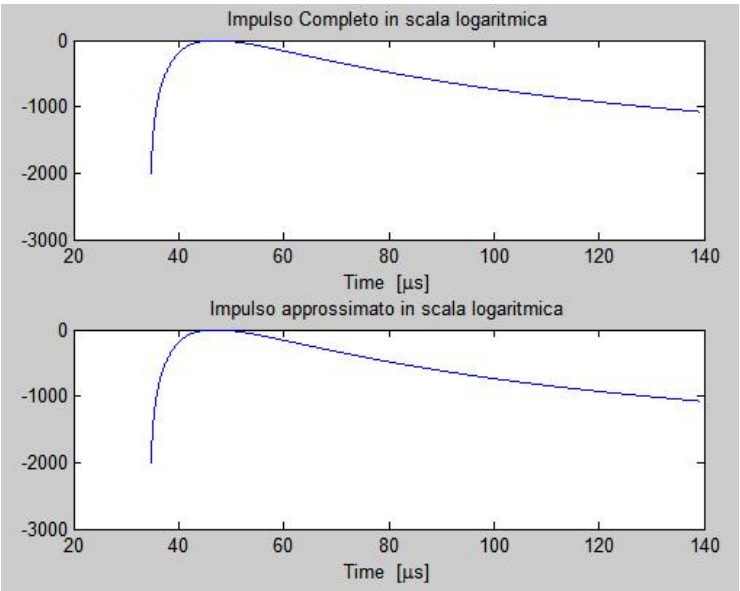

Figura 1.10 Rappresentazione, in scala logaritmica, dell'eco riflesso calcolato sia nella versione semplificata che nella versione completa

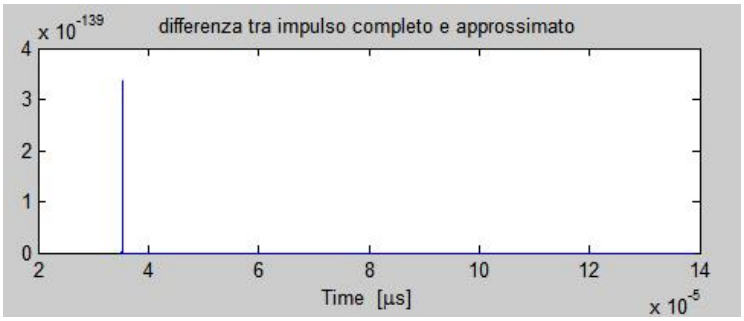

Figura 1.11 Differenza tra i campioni normalizzati in versione completa e quelli in versione semplificata

Nella Figura 1.11, viene mostrata la differenza tra i campioni normalizzati in versione completa e quelli nella versione semplificata. In questo caso, si può notare una differenza di solo $4*10^{-139}$, per pochissimi campioni. Ripetendo l'analisi, per vari valori della quota e dell'angolo di off-nadir, si può notare un trend peggiorativo nel caso in cui si aumenta il valore della quota e diminuisce il valore dell'angolo di off-nadir. Anche nel caso peggiore, con una quota di 8000m e angolo di off-nadir di 0,05°, si può individuare una differenza dello 0,008% sul valore massimo del segnale, che essendo normalizzato è pari ad uno. Attraverso queste considerazioni, la funzione erf(x) non influisce significativamente sul calcolo dell'impulso riflesso. Il calcolo dell'eco riflesso può essere implementato come nell'Equazione 1.29.

$$P_{r,norm} = \frac{IR^{Asym}}{\max(IR^{Asym})} = \frac{G(\varepsilon)}{\max(G(\varepsilon))}$$

Equazione 1.29

Potenza reale del segnale

Una volta elaborata e individuata la versione finale per il calcolo dell'eco riflesso normalizzato, è possibile procedere alla seconda parte per il calcolo della potenza reale del segnale. In questo caso, si parte dall'equazione del radar, ricavata dall'equazione del bilancio di radiocollegamento. Con il termine bilancio di radiocollegamento si indica la relazione formale che stabilisce il bilancio di potenza, in un sistema di telecomunicazione, tra la potenza ricevuta dal ricevitore in funzione di quella emessa dall'apparato trasmittente. Questa espressione include tutti i fattori di amplificazione e dissipativi lungo il canale in cui si propaga il segnale. La sua espressione generale è indicata nell'Equazione 1.30:

$$P_r = \frac{P_t\, G}{A(r)}$$

Equazione 1.30

Dove

- P_r rappresenta la potenza in ricezione,
- P_t rappresenta la potenza in trasmissione,
- G rappresenta il guadagno per eventuali amplificazioni del segnale prima o dopo il canale,

- A(r) rappresenta l'attenuazione, in funzione della distanza r, che il segnale subisce, o nel transito sul canale, o dovuta ad altri fattori di dissipazione di potenza, come disturbi fisici sul collegamento.

In altri termini, l'Equazione 1.30 afferma che la potenza ricevuta è uguale alla potenza emessa, a meno di un fattore al numeratore, che include tutti i guadagni, e un fattore al denominatore, che include tutte le attenuazioni e le perdite sul canale.

Nel caso di una singola superficie osservata, la quantità di potenza Pr che ritorna all'antenna ricevente è data dall'equazione del radar (Equazione 1.31). Questa equazione altro non è che l'equazione del bilancio di radiocollegamento applicata ad un sistema radar:

$$P_r = \frac{P_t G_t A_r \sigma}{(4\pi)^2 R_t^{\,2} R_r^{\,2} L}$$

Equazione 1.31

Dove

- G_t è il guadagno dell'antenna del trasmettitore,
- A_r è l'area equivalente di antenna del ricevitore,
- σ rappresenta la superficie equivalente dell'oggetto. Nel caso generale di target in moto essa rappresenta il valor medio, nel tempo, della superficie equivalente; si deve considerare il valor medio a causa della continua variazione di assetto dell'oggetto,
- R_t è la distanza del trasmettitore dall'oggetto,
- Rr è la distanza dell'oggetto dal ricevitore,
- L sono le perdite di attenuazione del mezzo atmosferico, dell'antenna e della catena ricevente.

Nel caso più comune, in cui l'antenna trasmittente e quella ricevente coincidono fisicamente, come nel nostro caso, si hanno alcune semplificazioni:

- La distanza del trasmettitore dall'oggetto è uguale a quella tra l'oggetto e il ricevitore, così Rt^2 Rr^2 può essere sostituito da R^4, dove R è la distanza dell'oggetto dal radar
- Il guadagno dell'antenna ricevente, espresso nell'Equazione 1.32, coincide con quella dell'antenna trasmittente
- L'area equivalente di antenna del ricevitore è uguale a quella del trasmettitore, $A_t=A_r=A$

$$G_t = G_r = \frac{4\pi}{\lambda^2} A$$

Equazione 1.32

Utilizzando le semplificazioni, l'Equazione 1.31 può essere espressa come:

$$P_r = \frac{P_t G_t^{\,2} \lambda^2 \sigma}{(4\pi)^3 R^4 L}$$

Equazione 1.33

L’Equazione 1.33 mostra come la potenza dell'onda riflessa diminuisce con la quarta potenza della distanza, quindi l'entità del segnale ricevuto è modesta, a fronte di una potenza trasmessa tipicamente elevata dell'impulso trasmesso.

Nel nostro caso, nell’Equazione 1.33 dobbiamo considerare:

- Pt pari a 30dBm, ossia 1W,
- Gt pari a 1,
- λ, la lunghezza d’onda della portante del radar, pari a 0.0084m,
- $\sigma = \sigma_0 * A_\sigma$, dove σ_0 è il vettore di backscattering, noto con l’angolo di off-nadir, e A_σ rappresenta l’area di scattering, mostrata nell’Equazione 1.34
- R = Slant Range SR
- L = 1

$$A_\sigma = \frac{(\pi \, SR \, \theta_{-3dB})^2}{\cos \xi}$$

Equazione 1.34

Attraverso queste considerazioni, si può riscrivere la Equazione 1.33 come:

$$P_{r,real} = \frac{10^{-7} P_{TX} \sigma_0 H^2}{SR^4}$$

Equazione 1.35

In questo caso, σ_0 verrà scelto tra alcuni valori precalcolati per vari valori dell’angolo di off-nadir.

Una volta calcolato $P_{r,real}$, è possibile calcolare l’espressione completa dell’eco riflesso utilizzando l’eco normalizzato, valutato nella prima parte con l’Equazione 1.29 e l’Equazione 1.25. L’espressione, per il calcolo dell’eco riflesso, è mostrato nell’Equazione 1.36.

$$P_r = P_{r,real} \cdot P_{r,norm}$$

Equazione 1.36

Dove P_r rappresenta l’eco reale riportato ai valori di potenza reali, $P_{r,real}$ il picco di potenza calcolato con l’Equazione 1.35 e $P_{r,norm}$, l’eco riflesso normalizzato.

La modulazione I&Q

Un'altra specifica richiesta per il sistema di test dell’RDA è la frequenza di campionamento, per la generazione del segnale, pari a 400Mhz. Per raggiungere questa frequenza, non implementabile direttamente in hardware, si utilizza una modulazione I&Q.

Questa modulazione permette di estrarre le due componenti del segnale, quella in fase e quella in quadratura. Le due componenti possono essere ricavate eseguendo una demodulazione in fase e in quadratura. Queste operazioni consistono in un prodotto, con $2\cos(2\pi f_0 t)$ e con $2\sin(2\pi f_0 t)$ come mostrato nell’Equazione 1.38 e nell’Equazione 1.39, che permettono di ottenere, attraverso un filtraggio passa basso, le componenti in fase e in quadratura del segnale. Una volta ottenute, le due

componenti del segnale possono essere utilizzate per ritrovare il segnale iniziale attraverso le operazioni mostrate nell'Equazione 1.37.

$$P_r = I_r \cdot \cos(2\pi f_0 t) + Q_r \cdot \sin(2\pi f_0 t)$$

Equazione 1.37

$$P_r 2\cos(2\pi f_0 t) = 2I_r \cos^2(2\pi f_0 t) + 2Q_r \sin(2\pi f_0 t)\cos(2\pi f_0 t)$$
$$= I_r + I_r \cos(4\pi f_0 t) + Q_r \sin(4\pi f_0 t)$$

Equazione 1.38

$$P_r 2\sin(2\pi f_0 t) = 2I_r \cos(2\pi f_0 t)\sin(2\pi f_0 t) + 2Q_r \sin^2(2\pi f_0 t)$$
$$= Q_r - Q_r \cos(4\pi f_0 t) + I_r \sin(4\pi f_0 t)$$

Equazione 1.39

Per chiarire la modulazione I&Q, viene mostrata la Figura 1.12, in cui si evidenziano le operazioni realmente eseguite sul segnale. Nella fase di demodulazione si separano le due componenti, indicando con $x_1(t)$ la componente in fase e con $x_2(t)$ la componente in quadratura. Attraverso un filtraggio passa basso finale, si ottengono le componenti I e Q. Nella fase di modulazione del segnale, le due componenti diventano delle modulanti per un seno ed un coseno che, poi sommate, permettono di ritornare al segnale iniziale.

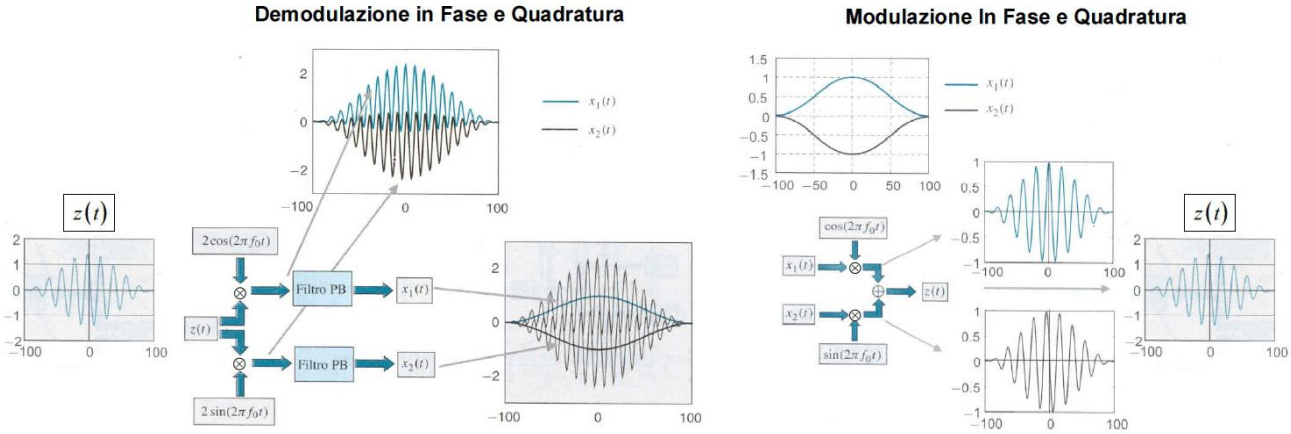

Figura 1.12 rappresentazione grafica delle operazioni di Demodulazione e Modulazione del segnale

In nostro caso, la modulazione I&Q viene utilizzata per ottenere una frequenza finale di 400Mhz. Attraverso un demodulatore, presente nell'RF Front End, si passa dalle due componenti I e Q, generate a 200Mhz dall'EG, al segnale, con un frequenza di campionamento di 400Mhz, richiesto in uscita. Quindi, una volta valutato l'eco riflesso, si devono determinare le componenti I e Q del segnale.

Le componenti I&Q, data la potenza, possono essere calcolate attraverso l'Equazione 1.40 e l'Equazione 1.41.

$$I = \sqrt{100P_r}\cos\left(\frac{4\pi}{\lambda}\cdot SR\right)$$

Equazione 1.40

$$Q = \sqrt{100P_r}\sin(\frac{4\pi}{\lambda} \cdot SR)$$

Equazione 1.41

Considerando λ, la lunghezza d'onda della portante del radar, di 0.0084m, possiamo riscrivere le espressioni delle componenti come nell'Equazione 1.42 e nell'Equazione 1.43.

$$I = \sqrt{100P_r}\cos(1{,}49895 \cdot 10^3 \cdot SR)$$

Equazione 1.42

$$Q = \sqrt{100P_r}\sin(1{,}49895 \cdot 10^3 \cdot SR)$$

Equazione 1.43

Algoritmo implementato nell'EG

L'elaborazione finale, ottenuta attraverso le semplificazioni presentate nel paragrafo precedente, dovrà essere implementata sull'unità digitale. L'algoritmo prevede due parti che saranno elaborate parallelamente. Una prima parte per il calcolo dell'eco riflesso, normalizzato, e una seconda per il calcolo della potenza reale del segnale e delle componenti I&Q per la generazione. La seconda parte, ossia il calcolo della potenza e delle componenti I&Q, verrà eseguito ogni PRI, ossia ad ogni impulso inviato dal radar RDA. Per la prima parte, che rappresenta il costo computazionale più alto, si utilizza un approccio differente. L'algoritmo per il calcolo dell'eco non viene eseguito ogni PRI, ma ogni ms. La valutazione, per applicare questo approccio, è stata puramente qualitativa ed è basata sull'approssimazione che, in 50msec, la dinamica è tale da non alterare significativamente la forma dell'eco. Questo è un buon compromesso tra precisione e velocità di calcolo. Nella Figura 1.13, viene presentata la strategia seguita per la programmazione dell'unità digitale.

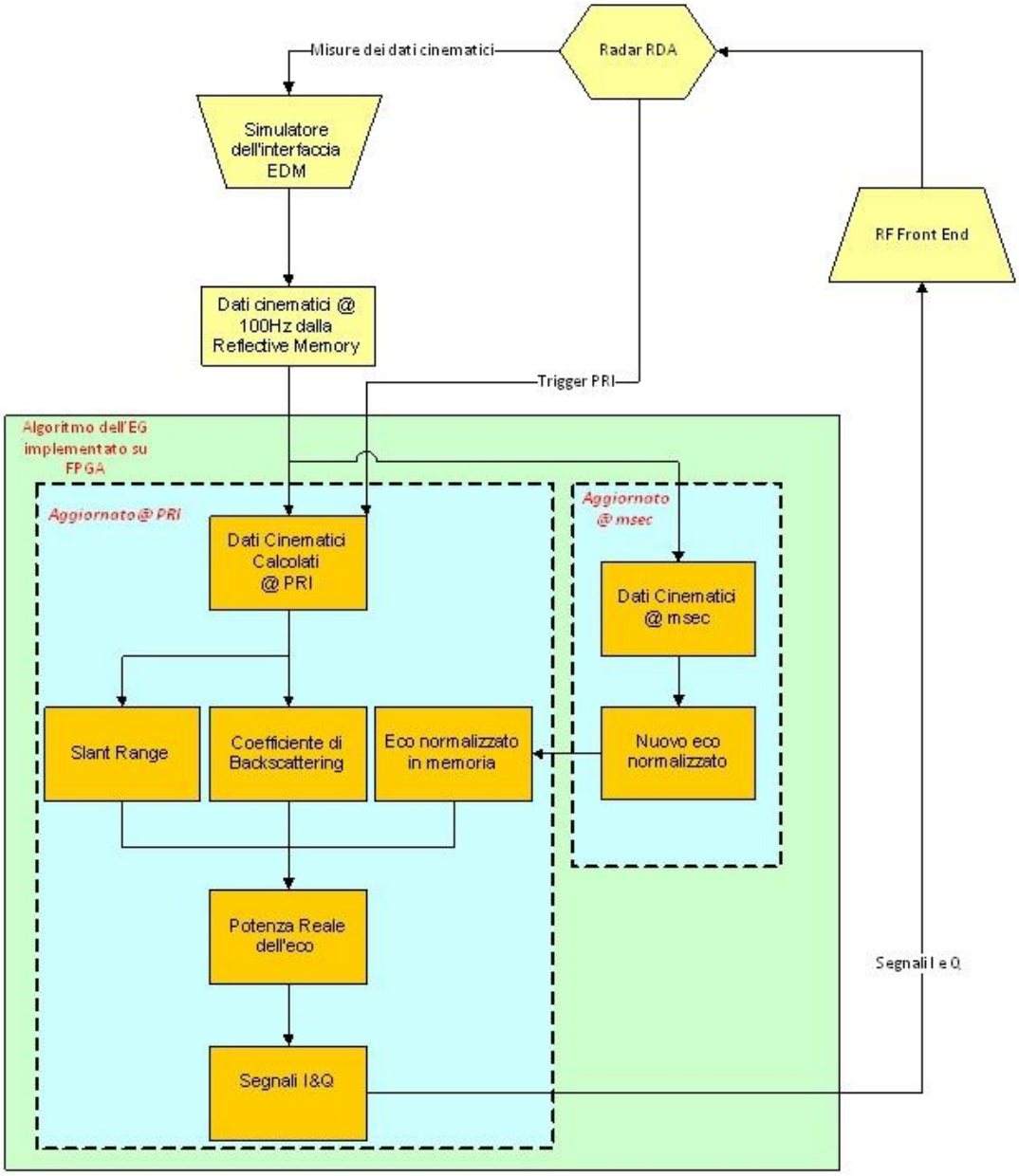

Figura 1.13 strategia implementativa dell'algoritmo, per il calcolo dell'eco riflesso, nell'unità digitale

Nella Figura 1.13, vengono evidenziati i due processi eseguiti, l'uno ad ogni PRI e l'altro ogni msec. Il primo processo è composto da quattro fasi. Inizialmente viene eseguito un'elaborazione dei dati cinematici. Questa operazione risulta necessaria perché i dati cinematici vengono aggiornati ogni 10msec, ossia un tempo molto più alto rispetto al periodo del PRI, tra 10 e 88μs. Per utilizzare i dati cinematici sempre aggiornati, viene eseguito l'aggiornamento ipotizzando un moto uniforme del radar. Per fare questo, il simulatore dell'interfaccia con il modulo EDM dovrà fornire, oltre ai dati riguardanti la posizione, anche i dati che si riferiscono alla velocità di spostamento del radar.

I dati ricevuti dal simulatore dell'interfaccia con il modulo EDM sono:

- X_{GNC} è la posizione in metri, sull'asse x, del modulo EDM
- Y_{GNC} è la posizione in metri, sull'asse y, del modulo EDM

- Z_{GNC} è la posizione in metri, sull'asse z, del modulo EDM
- $V_{x,GNC}$ è la velocità in m/sec, sull'asse x, del modulo EDM
- $V_{y,GNC}$ è la velocità in m/sec, sull'asse y, del modulo EDM
- $V_{z,GNC}$ è la velocità in m/sec, sull'asse y, del modulo EDM
- α_{GNC} è l'angolo di roll, in radianti, del modulo EDM
- β_{GNC} è l'angolo di pitch, in radianti, del modulo EDM
- γ_{GNC} è l'angolo di yaw, in radianti, del modulo EDM
- $\dot{\alpha}_{GNC}$ è la velocità angolare di roll, in rad/sec, del modulo EDM
- $\dot{\beta}_{GNC}$ è la velocità angolare di pitch, in rad/sec, del modulo EDM
- $\dot{\gamma}_{GNC}$ è la velocità angolare di yaw, in rad/sec, del modulo EDM

Il sistema di riferimento utilizzato per valutare i parametri è mostrato in Figura 1.14.

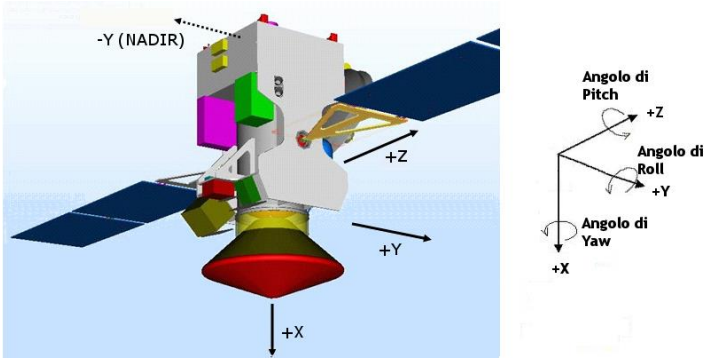

Figura 1.14 sistema di riferimento per il calcolo dei dati cinematici.

Le espressioni implementate per eseguire l'aggiornamento dei dati cinematici sono mostrate nell'Equazione 1.44 e nell'Equazione 1.45.

$$\begin{cases} X = X_{GNC} + V_{x,GNC} \cdot (t_{PRI} - t_{GNC}) \\ Y = Y_{GNC} + V_{y,GNC} \cdot (t_{PRI} - t_{GNC}) \\ Z = Z_{GNC} + V_{z,GNC} \cdot (t_{PRI} - t_{GNC}) \end{cases}$$

Equazione 1.44

$$\begin{cases} \alpha = \alpha_{GNC} + \dot{\alpha}_{GNC} \cdot (t_{PRI} - t_{GNC}) \\ \beta = \beta_{GNC} + \dot{\beta}_{GNC} \cdot (t_{PRI} - t_{GNC}) \\ \gamma = \gamma_{GNC} + \dot{\gamma}_{GNC} \cdot (t_{PRI} - t_{GNC}) \end{cases}$$

Equazione 1.45

Nell'Equazione 1.44 e nell'Equazione 1.45, le variabili X, Y, Z, α, β e γ rappresentano i valori aggiornati dei dati cinematici. Ogni 10msec, i dati cinematici verranno aggiornati dal simulatore dell'interfaccia con il modulo EDM, sostituendo i valori presenti nelle espressioni con valori aggiornati. Questi parametri sono trasmessi all'EG, ad alta velocità, tramite interfaccia reflective memory. La seconda fase, mostrata nella Figura 1.13, consiste nel valutare dei parametri necessari per le fasi successive, quali lo slant range e il coefficiente di backscattering. Un'altra operazione

eseguita in questa fase è di aggiornare, se necessario, o andare a leggere in memoria l'eco normalizzato, calcolato in precedenza, per le fasi successive. L'operazione di lettura, o aggiornamento, è necessaria perché la frequenza di aggiornamento dell'eco è differente dalla frequenza di esecuzione del processo. La terza fase consiste nel calcolo della potenza reale, attraverso l'Equazione 1.35 e l'Equazione 1.36. Nell'ultima fase del processo vengono calcolate le componenti I&Q del segnale per la generazione con l'Equazione 1.42 e l'Equazione 1.43.

Nel processo eseguito in parallelo, ogni msec, sarà calcolato e aggiornato il valore dell'eco normalizzato. Anche in questo caso sono utilizzati i dati cinematici aggiornati a ogni PRI per maggiore precisione. Una volta calcolati, i campioni dell'eco normalizzato saranno memorizzati per le letture del processo eseguito a ogni PRI.

2 . La piattaforma Hardware

Il Sistema PXI-e

L'hardware utilizzato per la realizzazione del sistema di test si basa su una piattaforma PXI (PCI eXtensions for Instrumentation). Questa piattaforma, basata su PC, è in grado di offrire soluzioni per i sistemi di misura e automazione ad elevate prestazioni e costi ridotti. Questa piattaforma combina le funzioni di bus elettrico PCI (Peripheral Component Interconnect) con il pacchetto di CompactPCI e aggiunge bus di sincronizzazione specializzati.

PXI, inoltre, dispone di caratteristiche meccaniche, elettriche e software che permettono di realizzare sistemi completi di test e misura e acquisizione dati.

Dal 2005, la piattaforma PXI è stata potenziata con l'introduzione della tecnologia PXI Express, che aumenta l'ampiezza di banda dei sistemi PXI da 132MB/s a 6GB/s, con un incremento di quarantacinque volte, mantenendo la compatibilità software e hardware con i moduli PXI.

Quindi la piattaforma PXI express è un'ottima scelta per la realizzazione del sistema di test del radar RDA. Una rappresentazione generica di un sistema PXI è mostrata in Figura 2.1.

I sistemi PXI sono costituiti da tre elementi fondamentali:

- Chassis
- Controller di sistema
- Moduli periferici

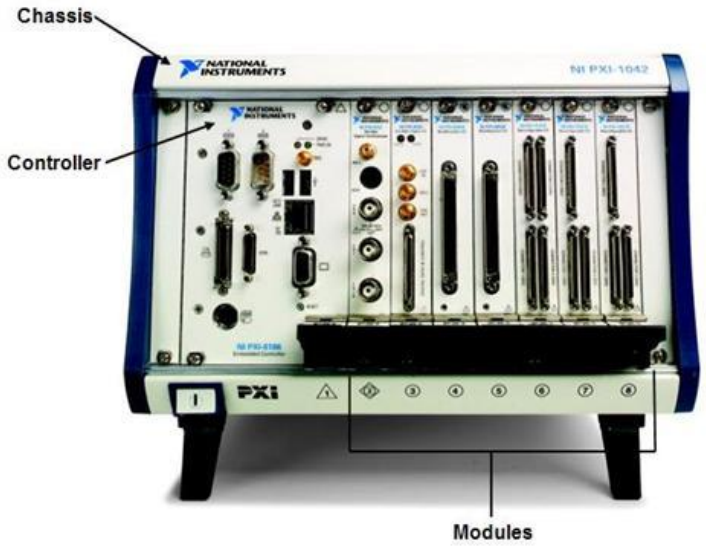

Figura 2.1 Chassis PXI standard contenente un
controller di sistema embedded e sette moduli periferici

Chassis PXIe-1082

Lo chassis è l'alloggiamento fisico per il sistema e permette l'utilizzo dei moduli periferici PXI e PXI Express, degli alimentatori AC/DC e di strutture hardware per il condizionamento integrato del segnale. Alcuni chassis dispongono di slot ibridi e slot periferici che consentono di utilizzare entrambi i moduli PXI e PXI Express. Gli chassis possono essere configurati in maniera personalizzata, grazie alla sua natura modulare, per far fronte alle diverse esigenze applicative.

All'interno dello chassis è presente il backplane PXI, ad elevate prestazioni. Esso include il bus PCI, i bus di triggering e temporizzazione. Nel backplane viene generato un clock di riferimento di sistema dedicato a 10MHz,per la sincronizzazione dell'intero sistema; è presente un PXI trigger bus e star trigger bus, per la propagazione di segnali di trigger tra i vari moduli e un bus locale slot-to-slot, per una temporizzazione avanzata. Questi permettono una sincronizzazione e una comunicazione in banda, mantenendo inalterate le prestazioni dello standard PXI.

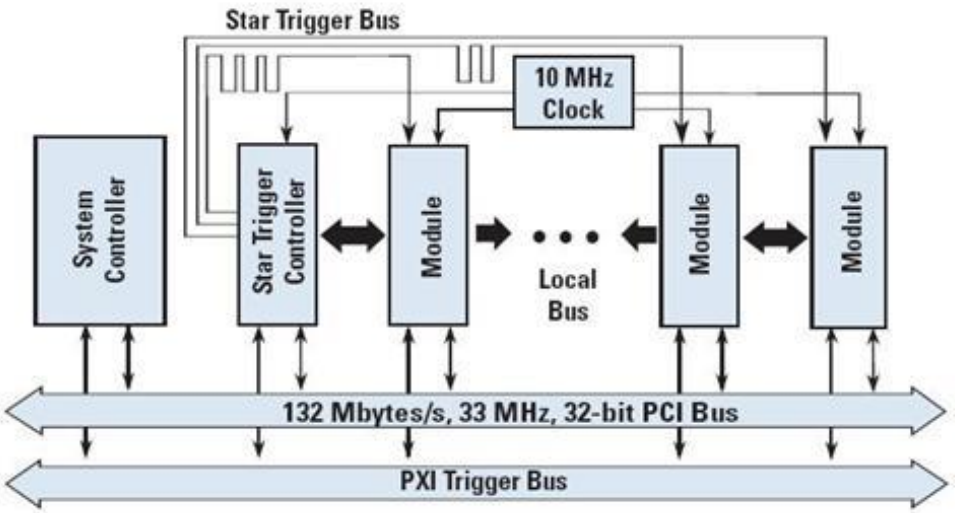

Figura 2.2 Bus PXI di temporizzazione e triggering

Sulla base delle funzionalità PXI, PXI Express aggiunge nuove connessioni elettriche per la temporizzazione e sincronizzazione, con l'inserimento di un clock di sistema differenziale a 100 MHz, un segnale differenziale e uno star trigger differenziale.

Mediante clocking e sincronizzazione differenziale, i sistemi PXI Express consentono di trasmettere i clock di strumentazione a frequenze maggiori e di avere una migliore immunità al rumore.

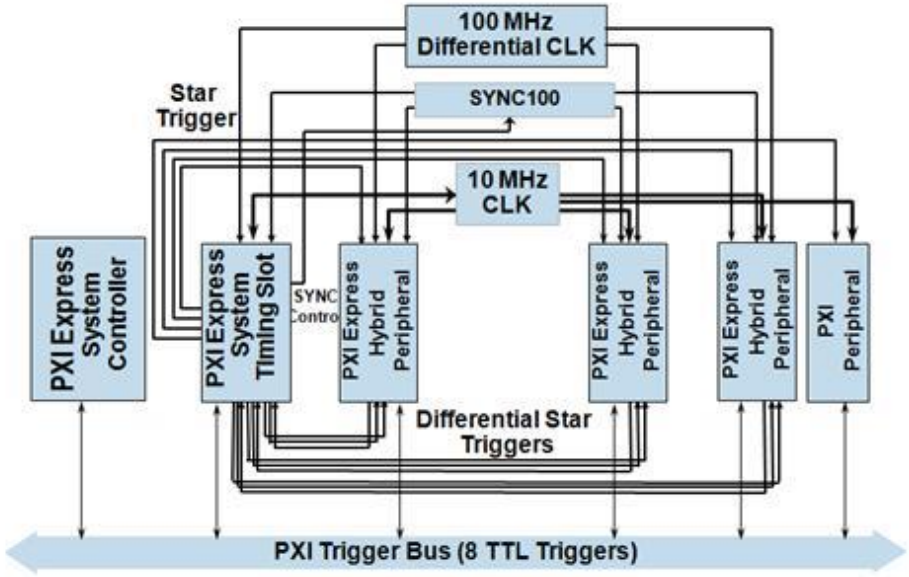

Figura 2.3 I bus di temporizzazione e triggering PXI Express estendono le funzionalità PXI e aggiungono un clock di sistema differenziale, elaborazione del segnale differenziale e star trigger differenziale sul backplane,

Grazie a questi bus di temporizzazione e di triggering, è possibile sviluppare, come nel nostro caso, sistemi che richiedono una sincronizzazione precisa.

Lo chassis utilizzato nel sistema è il PXIe-1082, vale a dire uno chassis PXI Express con un backplane per i collegamenti ad alta velocità, in cui è possibile utilizzare i vari moduli PXI.

Il PXIe-1082 è compatibile con i moduli PXI Express, e, con l'utilizzo di slot ibridi, con gli standard PXI e CompactPCI. Sono presenti 3 PXI Express slot e 4 PXI slot, che risultano compatibili sia con un modulo di tipo PXI Express sia con un modulo PXI Standard.

Questo chassis include un Clock build-in di 10MHz di riferimento, un bus per il PXI trigger, un PXI Star trigger, un clock da 100MHz, un bus SYNC100 e un PXI Star trigger differenziale per la temporizzazione e la sincronizzazione del sistema.

Lo chassis PXI Express dispone di quattro PCI Express Link (ognuno dei quali consente un traffico dati di 1 GB/s in ogni direzione) per la gestione dei collegamenti tra i vari moduli. Come mostrato in Figura 2.4, un PCI Express Link è collegato direttamente con lo slot 2, mentre gli altri tre sono collegati ad altrettanti switch, ciascuno dei quali può essere connesso con due slot PXI Express periferici. Ogni slot ha a disposizione un collegamento ad alto datarate (1GB/s per ogni direzione). Nel caso in cui siano utilizzati due slot collegati a uno stesso switch, i moduli PXI Express condivideranno la banda di 1GB/s totale del PCI Express Link. Per i quattro slot ibridi è integrata un'interfaccia PCIe/PCI Bridge, che permette di utilizzare sia il PCI Express Link, sia lo standard PCI bus a 32bit e 33MHz implementato nel backplane.

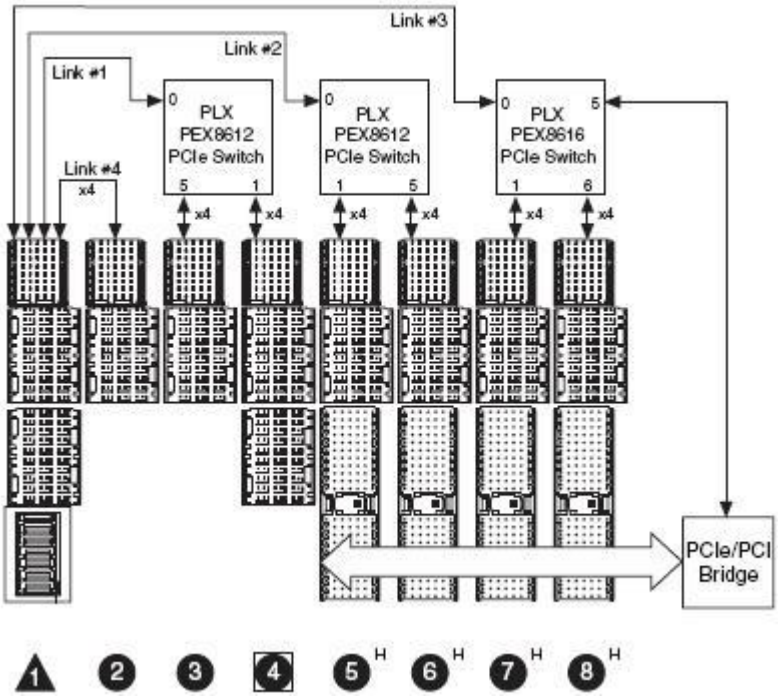

Figura 2.4 I bus per la comunicazione tra i moduli PXI Express

Per la temporizzazione, il backplane del PXIe-1082 implementa il clock di 10MHz e il clock differenziale di 100MHz che saranno utilizzati per la sincronizzazione del sistema. Lo chassis PXIe-1082 include un connettore IN/OUT BNC con cui è possibile collegare un clock a 10MHz per sincronizzare il sistema con un clock esterno.

Controller PXIe-8133

Il controller PXI svolge la funzione di controllo della struttura. All'interno del controller è installato il software per la programmazione e l'utilizzo dei moduli PXI periferici. La maggior parte degli chassis PXI presenta un alloggiamento apposito per tale elemento, normalmente il primo slot a sinistra.

Varie soluzioni sono previste per il controller di sistema come, per esempio, una configurazione prevede il controllo remoto dei moduli PXI da desktop, workstation, server o laptop. Un'altra soluzione prevede l'utilizzo di un controller embedded a elevate prestazioni, cioè un vero e proprio pc, con sistema operativo Microsoft (Windows 7/XP), senza la necessità di un PC esterno per il controllo del sistema. La configurazione hardware, utilizzata nel sistema di test, include un Controller Embedded PXI.

I controller embedded sono pc standard dotati di CPU integrata, disco fisso, RAM, Ethernet, scheda video, tastiera/mouse, USB e altre periferiche con sistema operativo Microsoft Windows.

Nel sistema utilizzato, per lo sviluppo della tesi, è presente il NI PXIe-8133, un controller embedded a avanzate prestazioni, basato su processore Intel Core i7-820QM.

Figura 2.5 Il controller NI PXI-8133 è dotato di processore quad-core a 3.06 GHz Intel Core i7-820QM.

Per sfruttare al meglio le potenzialità dei quattro core presenti su NI PXIe-8133, le applicazioni vengono progettate con quattro thread di esecuzione indipendenti, implementando strategie di programmazione, come, il parallelismo di task e di dati. Per incrementare il numero di thread a disposizione, il processore utilizza la tecnologia Hyper-threading di Intel, che permette di suddividere ciascuno dei quattro core in due core virtuali, così da poter disporre di otto core virtuali in totale.

Per le applicazioni che non sono programmate per i processori multicore, Intel ha introdotto la tecnologia Turbo Boost. Tale tecnologia consente, infatti, di incrementare la frequenza di base del processore, che nel PXIe-8133 è di 1.73Ghz, fino a 3,06Ghz portando in idle i core non utilizzati, incrementando la velocità di esecuzione dei processi.

Il controller PXI-8133 dispone di quattro collegamenti "PCI Express Gen 2 lanes" verso il backplane dello chassis PXI. In tal modo, per il controllo dei moduli periferici, il controller ha un bus fino a 2GB/s di larghezza di banda dedicata per ogni slot e un'ampiezza di banda totale di 8GB/s.

Data l'elevata integrazione del sistema, si utilizza un diagramma a blocchi per individuare le diverse interfacce e le varie connessioni tra i moduli.

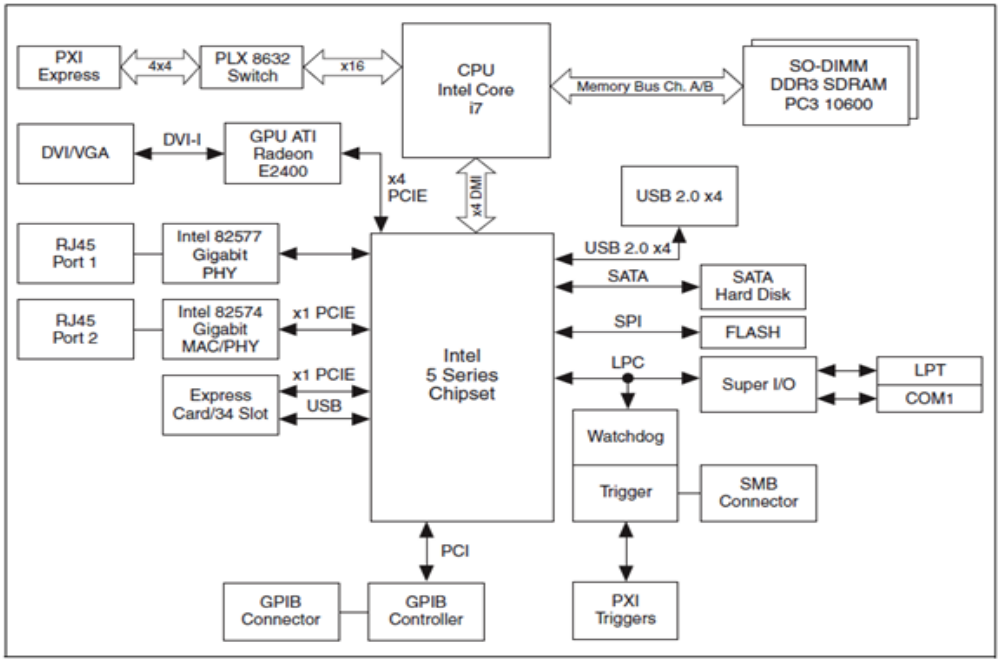

Figura 2.6 Diagramma a blocchi del NI PXIe-8133

Il NI PXIe-8133 utilizza un Chipset Intel-5. Quest'ultimo, attraverso un x4 DMI, crea un'interfaccia tra il processore Intel Core I7 e altri moduli che forniscono le uscite standard di un PC. Il PXIe-8133 usa una SDRAM DDR3 Dual Channel a 1333MHz, che consente l'utilizzo di questo controller in applicazioni che richiedono elevata elaborazione e analisi dei dati.

Il PXIe-8133 presenta i seguenti blocchi logici:

- FPGA socket 189, alloggio per il processore Intel i7
- Il processore che mette in connessione la SDRAM, lo Switch PLX8632 per i moduli PXI Express e il Chipset Intel-5
- Un blocco SO-DIMM, che prevede due socket per SDRAM DDR3 a 64bit, ciascuno dei quali contiene un banco di RAM da 2GB
- SMB e PXI Express trigger, che permettono una connessione riconfigurabile del trigger con connessione SMB sul pannello frontale
- Il blocco Watchdog timer, supervisore per la temporizzazione, che consente di resettare il controller o generare dei segnali di trigger
- Il chipset Intel-5 che svolge il ruolo di collegamento con PCI, USB, Serial ATA, Express Card e PXI Express
- La connessione Serial ATA, che permette il collegamento con l'hard disk da 120GB, con velocità di trasferimento fino a 150MB/s.
- Il PXI Express controller che permette di connette il PXIe-8133 con il PXI Express/CompactPCI Express situato sul backplane

Moduli PXI

I moduli PXI permettono di aggiungere, al sistema, degli elementi per far fronte alle diverse esigenze applicative. Questo standard offre una larghezza di banda elevata e minore latenza con I/O modulare. Ci sono moduli PXI per la strumentazione, l'acquisizione di dati, la sincronizzazione avanzata, l'interfaccia con altri bus, input e output analogici, input e output digitali, l'elaborazione digitale dei segnali e dei generatori di segnali per la realizzazione dei sistemi.

Trattandosi di uno standard industriale aperto, sono disponibili oltre 1.500 moduli da oltre settanta produttori. Giacché PXI è compatibile con lo standard CompactPCI, è possibile utilizzare moduli CompactPCI su sistemi PXI o PXI Express.

PXIe-7965R per FPGA

Il modulo PXIe-7965R, presente nel sistema, appartiene alla famiglia NI FlexRio, che permette di utilizzare delle FPGA (Field-Programmable gate array) all'interno di sistemi PXI Express. Questi moduli sono spesso associati a dei moduli I/O adapter, per creare sistemi I/O riconfigurabili su piattaforma PXI, ad elevate prestazioni.

I moduli FlexRIO Adapter Module I/O diventano necessari, nel caso in cui si debbano implementare delle applicazioni su FPGA con un'interfaccia analogica. Tali moduli devono essere programmati utilizzando il software NI Labview FPGA Module.

Figura 2.7 Modulo PXIe-7965R e modulo I/O Adapter NI5781 in cui vengono visualizzati i circuiti integrati e i collegamenti tra i due moduli

Il modulo PXI Express 7965R utilizza una FPGA Xilinx Virtex-5 SX95T. Sul modulo è presente, inoltre, una DRAM da 512MB, cui si può accedere con un collegamento ad alta velocità di 3.2GB/s.

La Virtex-5 SX95T è ottimizzata per un'elaborazione dei segnali digitali, ad alta velocità, con 640 DSP slice. Queste slice permettono di eseguire le funzioni di moltiplicazione e filtraggio in un unico periodo di clock.

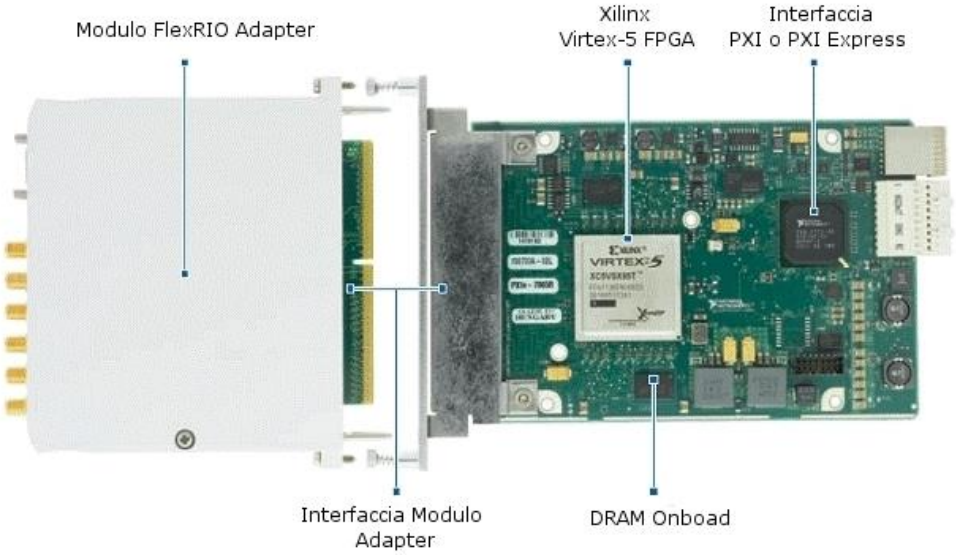

Figura 2.8 Elementi integrati nel modulo PXIe-7965R e il modulo I/O Adapter

La PXI Express 7965R dispone di alcuni circuiti integrati, gli NI STC-3, utilizzati per ottimizzare il collegamento con i PCI Express x4 Link, presenti nel backplane dello chassis. Questi ASIC riducono i tempi necessari per la comunicazione con il processore e consentono un trasferimento di tipo Peer-to-Peer tra due moduli.

FPGA Virtex-5 SX95T

Le FPGA fanno parte della classe dei dispositivi riconfigurabili. Questi sistemi possono essere configurati per implementare, in un numero pressoché illimitato di volte, funzionalità di complessità variabile.

La loro configurazione è generalmente memorizzata su una memoria non permanente, per cui il dispositivo ha la necessità di essere inizializzato all'accensione. La flessibilità della FPGA le rende una piattaforma ottimale per la programmazione general purpose.

La struttura di base di una FPGA è rappresentata nella Figura 2.9.

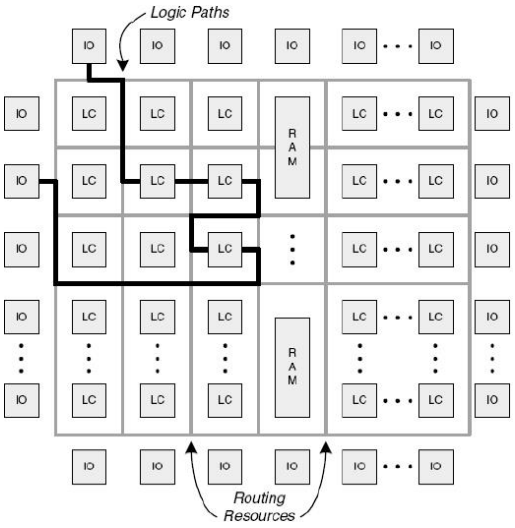

Figura 2.9 Struttura di una FPGA

Le funzionalità logiche del dispositivo sono fornite da una matrice di Slice, con la possibilità d'interconnessione tra loro.

Ogni slice possiede una Look-Up Table (LUT), ossia una memoria RAM a n ingressi e una o più uscite, che permette di realizzare una qualunque funzione logica di n variabili. All'interno di una slice sono, inoltre, presenti elementi di memoria (flip-flop D) e d'instradamento dei dati (multiplexer).

Una rappresentazione molto semplificata di una slice è quella rappresentata in Figura 2.10.

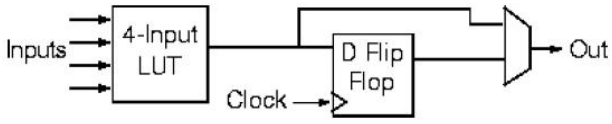

Figura 2.10 Rappresentazione di una Slice semplificata

Ogni slice può essere connessa ad altre mediante switch box, posti alle intersezioni delle risorse di routing, come mostrato in Figura 2.9. Tutte le connessioni sono configurabili, garantendo così, la massima flessibilità del dispositivo.

Nello stesso modo sono configurabili le interconnessioni con gli I/O pad, che forniscono il mezzo di comunicazione del chip con l'esterno permettendo, ad esempio, la scelta dello standard elettrico da utilizzare.

Le FPGA sono inoltre dotate anche di altre risorse configurabili come clock manager, blocchi di memoria RAM e moltiplicatori hard-wired.

Un esempio di FPGA è la Xilinx Virtex-5 SX95T, presente nella PXIe-7965R. Si tratta di un dispositivo FPGA orientato all'elaborazione dei segnali e con numerosi canali per il trasferimento di dati seriali ad alta velocità.

La famiglia Virtex-5 è caratterizzata da CLB (Configurable Logic Block), basate su delle LUT a 6 ingressi, che provvedono alla logica combinatoria e a quella sequenziale. Nelle CLB sono presenti, inoltre, dei blocchi di memoria distribuita e delle risorse di Shift Register.

I moduli di blocchi di RAM, distribuiti sull'FPGA, sono dei moduli dual port da 36Kbit che possono essere uniti per la realizzazione di una memoria più ampia. Questi blocchi di RAM, grazie ad una logica interna dedicata, possono essere utilizzati per implementare una coda FIFO programmabile.

I blocchi di I/O creano un'interfaccia tra i pin del package e la logica riconfigurabile interna. Gli IOBs (IO Block) utilizzati supportano gli standard di uscita più diffusi.

La Virtex-5 include, al suo interno, delle slice DSP48E con un moltiplicatore 25x18 e un sommatore/sottrattore/accumulatore a 48bit per l'elaborazione digitale dei segnali.

I blocchi CMT (Clock Managment Tile) forniscono le risorse per la distribuzione del clock all'interno dell'FPGA. Ogni CMT contiene un blocco PLL e 2 DCM (Digital Clock Managment) auto calibranti.

Nei dispositivi della famiglia Virtex-5 sono contenuti dei blocchi Endpoint integrati per la programmazione con l'interfaccia PCI Express, compatibile con lo standard x1, x4, e x8 PCI Express Endpoint.

Questi endpoint permettono all'FPGA di interagire con altri moduli attraverso i collegamenti PCI, PXI e PXI Express. Questa compatibilità è necessaria per utilizzare la Virtex-5 SX95T all'interno di sistemi PXI Express, come nel nostro caso.

La matrice generale di routing (GRM) fornisce un insieme di switches, per il routing tra i vari elementi. Ogni elemento programmabile è collegato alla matrice degli switch permettendo delle connessioni multiple con la matrice generale di routing.

Tutti gli elementi, come le LUT e GRM, sono controllati da valori memorizzati in elementi di memoria statica. Questi valori sono caricati nell'FPGA durante l'avvio e la configurazione. Questi ultimi possono essere modificati per cambiare le funzioni degli elementi programmabili.

CLB

I CLB sono le risorse logiche per implementare circuiti sequenziali e combinatori. Ogni CLB è connesso alla matrice delle interconnessioni, per l'accesso alle risorse di routing. Ognuna contiene al suo interno una coppia di slice.

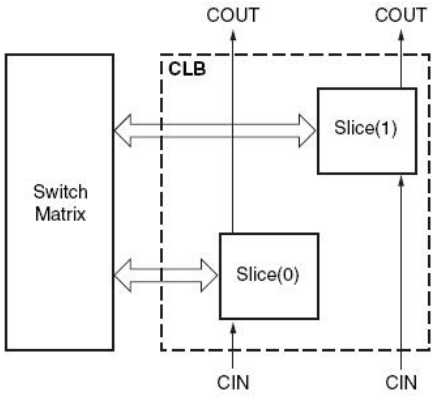

Figura 2.11 Rappresentazione di un CLB

Queste due slice, in un unico CLB, non hanno delle connessioni dirette, ma sono organizzate in colonne. Ogni slice ha una catena di carry da poter utilizzare nell'implementazione dei circuiti.

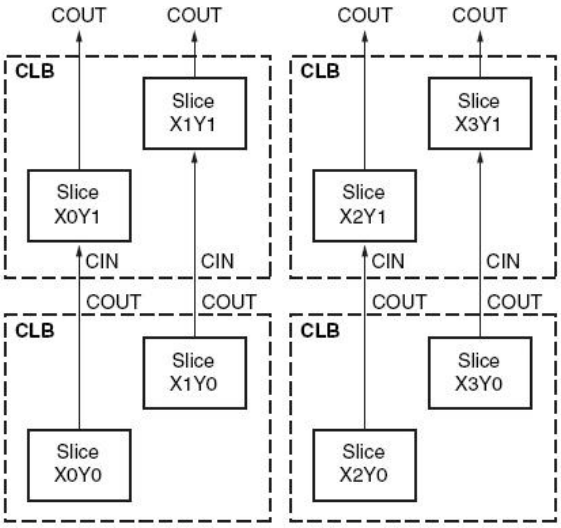

Figura 2.12 Struttura a colonne delle slice all'interno dei CLBs con le catene di carry

Ogni slice contiene quattro LUT, quattro elementi di memorizzazione, la logica per il riporto e dei multiplexer per aumentare la logica implementabile. Alcune slice possono utilizzare la RAM distribuita come elementi di memorizzazione, per la realizzazione di registri a scorrimento a 32 bit.

Le slice che supportano queste funzioni addizionali sono chiamate SLICEM, le altre che non le implementano vengono indicate come SLICEL.

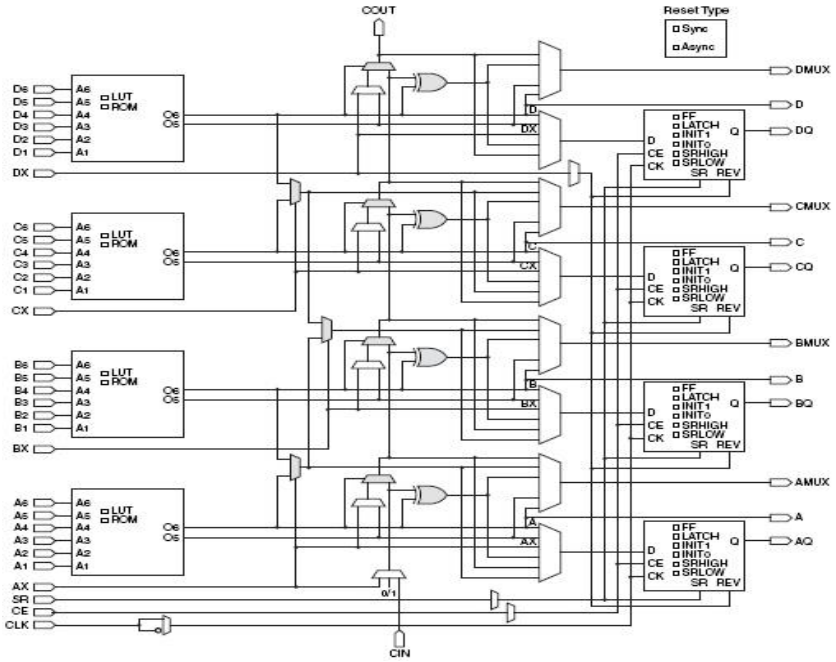

Figura 2.13 Schema di una SLICEL

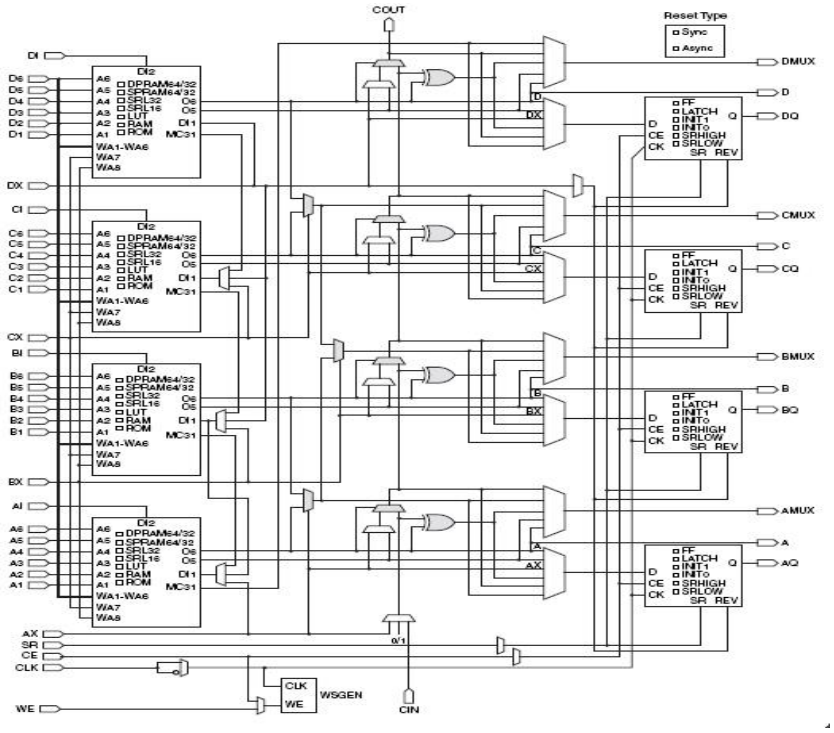

Figura 2.14 Schema di una SLICEM

Per la Virtex-5 SX95T si ha a disposizione di un insieme di CLB distribuito in 160 righe e 46 colonne, con 58880 LUT a 6 input.

La RAM distribuita sul dispositivo è di 1560Kbit. La memoria è distribuita in 58880 Flip Flop per l'elaborazione sequenziale dei segnali.

Quindi, in ogni CLB sono presenti 2 slice composte da 8 LUT e 8 Flip Flop con 2 catene aritmetiche e per il riporto.

Blocchi di RAM

I blocchi di RAM nella famiglia Virtex-5 hanno a disposizione 36Kbit. Questi blocchi di memoria possono essere riconfigurate in varie modalità per aumentare le possibili applicazioni; possono essere utilizzati come:

- 64Kx1 (con due moduli collegati in cascata),
- 16Kx2,
- 8Kx4,
- 2Kx18,
- 1Kx36.

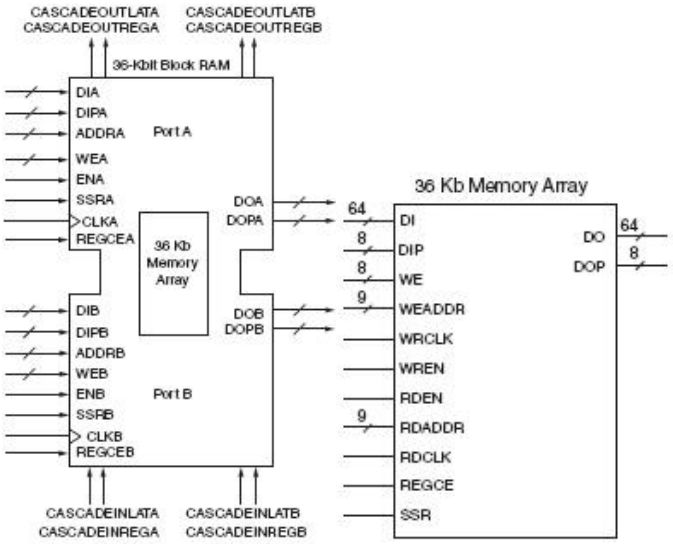

Figura 2.15 A sinistra il vero Data Flow e a destra quello semplificato di un blocco di RAM dual-port

I blocchi di RAM, utilizzati nelle Virtex-5, possono eseguire una scrittura o una lettura sincrona con il rispettivo clock. Dispongono di porte d'ingresso e d'uscita completamente indipendenti, con la possibilità di implementare una grandezza differente tra le porte di lettura e scrittura.

Durante un'operazione di scrittura la memoria può essere impostata in due modalità differenti: può dare in uscita il dato modificato, riflettendo il nuovo dato che si sta scrivendo, o dare in uscita il dato precedente che si sta sovrascrivendo.

Nella SX95T, i blocchi di RAM implementano una logica dedicata per semplificare la realizzazione di una coda FIFO sincrona o multirate (asincrona).

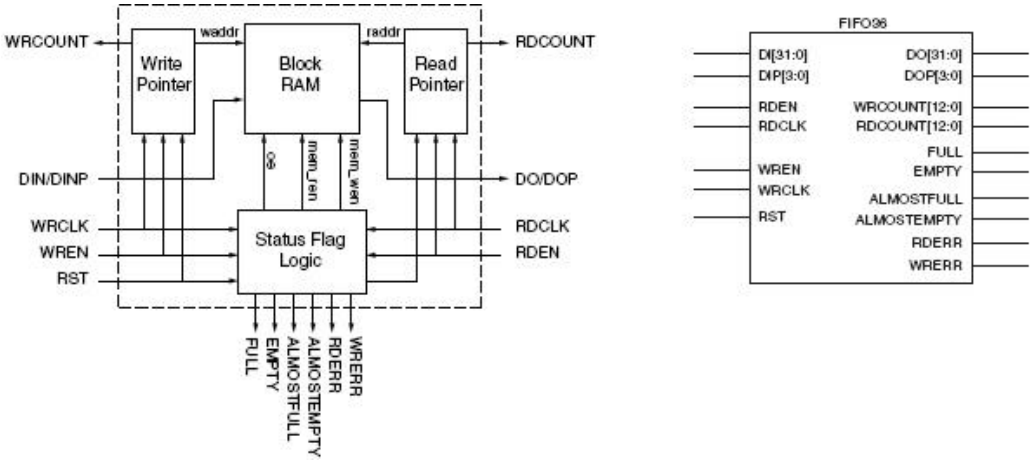

Figura 2.16 A sinistra una rappresentazione di alto livello dell'architettura FIFO

Implementata e a destra una primitiva per l'uso di una FIFO36

Questo elimina la necessità, per realizzare una FIFO, di logica addizionale su CLB per contatori, comparatori e per la generazione di flag di stato, rendendo possibile la realizzazione della FIFO solo

con dei blocchi di RAM. Per l'implementazione, una porta del blocco di RAM sarà usata come porta per la lettura della FIFO, l'altra come porta per la scrittura. Il dato sarà letto e scritto sul fronte di salita del clock. Può essere utilizzato un clock comune o indipendente per la lettura e la scrittura, rendendo possibile eseguire le due operazioni in istanti differenti.

L'operazione di scrittura è sincrona al clock WRCLK. La word, disponibile nella porta DI della FIFO, verrà memorizzata, ogni volta che il segnale WREN è alto, per un tempo di set-up prima del fronte di salita del segnale di clock WRCLK. L'operazione di lettura viene eseguita in maniera analoga. In questo caso, il dato in lettura, nella porta DO, sarà disponibile quando il segnale RDEN sarà alto un tempo di set-up prima del fronte di salita del segnale di clock RDCLK.

Il controllo del data-flow è automatico, l'utente non deve occuparsi della sequenza degli indirizzi di lettura e scrittura, anche se le porte WRCOUNT e RDCOUNT sono accessibili, per essere utilizzati in particolari applicazioni. Per la programmazione e per l'uso della RAM come FIFO, sono presenti dei flag di EMPTY, FULL e dei flag di avvertimento ALMOSTEMPTY e ALMOSTFULL per evitare errori in lettura e scrittura.

Xtreme DSP Slice

Le Xtreme DSP slice (chiamate anche slice DSP48E) sono le unità DSP per l'implementazione efficiente dell'elaborazione di segnali. Queste slice possono lavorare a frequenze di clock di 550MHz.

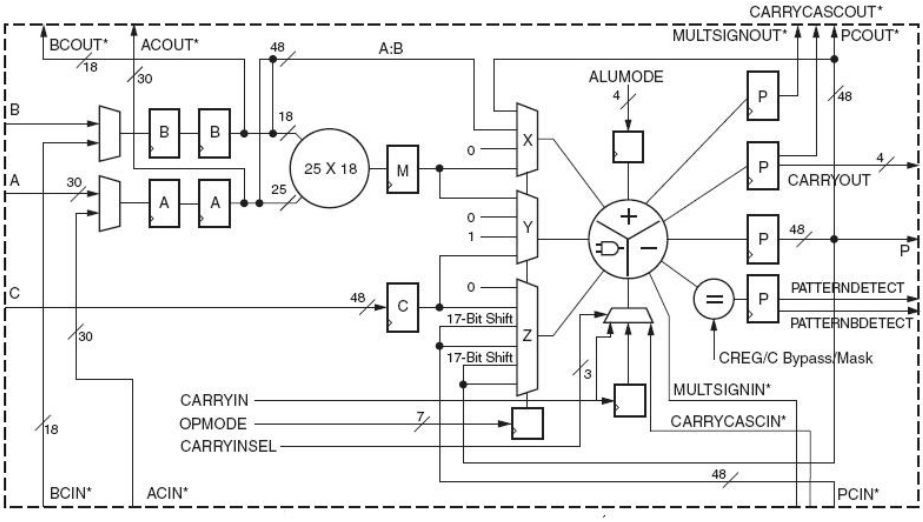

Figura 2.17 Diagramma a blocchi di una DSP48E slice della Virtex-5

Come si può osservare dalla Figura 2.17, ogni DSP48E contiene un moltiplicatore dedicato, a due ingressi 25x18 bit, e un sommatore/sottrattore/unità logica a 48bit che può agire su operatori senza segno o in complemento di due. Queste risorse, unite con dei registri e della logica combinatoria presente nel DSP, permettono di implementare, in maniera automatica, dei contatori e degli accumulatori. Tutti i moduli aritmetici sono hard-wired, ossia non sono implementati in logica riconfigurabile, ma in una configurazione fissa. L'interconnessione tra differenti slice DPS48E e con la logica dell'FPGA è, tuttavia, riconfigurabile.

Per implementare efficacemente le elaborazioni di segnali più avanzate, i DSP48E sono distribuiti in colonne. Le slice sono compatibili con il collegamento in cascata, perché i due datapath ACOUT, BCOUT e le uscite della slice (PCOUT, MULTSIGNOUT, e CARRYCASCOUT) sono presenti nella parte superiore della slice. Per gli ingressi, inoltre, è possibile scegliere tra due percorsi, uno proveniente dalla parte inferiore e uno da destra, rappresentati nel diagramma a blocchi, per selezionare la propagazione, o non propagazione dei dati. Questa configurazione è vantaggiosa nel caso in cui si devono realizzare dei filtri digitali, poiché non vengono utilizzate le risorse di routing generali, ma i collegamenti in cascata, per la propagazione dei dati. Quindi le slice DSP48E, all'interno delle FPGA della famiglia Virtex-5, sono organizzate in colonne. Questa differente organizzazione, rispetto alle Virtex-4 in cui i DSP sono distribuiti in righe, è il maggiore miglioramento apportato per i DSP nelle Virtex-5. I differenti approcci implementativi sono mostrati nella Figura 2.18.

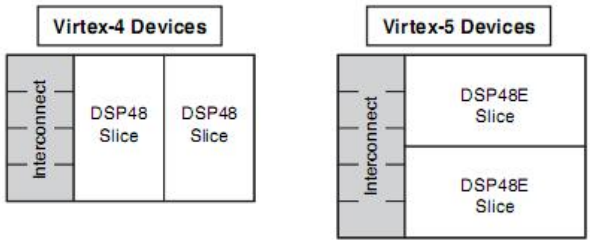

Figura 2.18 Organizzazione in colonne per i DSP48E Slice

Ogni DSP48E è disposta orizzontalmente con un blocco di RAM, in modo che la memoria distribuita, nelle elaborazioni, possa essere utilizzata con il minore tempo di latenza. Questo risultato è ottenuto perché i blocchi di RAM sono più vicini ai centri di elaborazione e accessibili con più facilità.

Le risorse presenti in una slice DSP consistono in:

- un moltiplicatore, in complementi di due, 25x18bit
- tre multiplexer a 48 bit, per selezionare gli operandi
- un addizionatore/sottrattore a tre input o un'unità logica a due ingressi.

Il datapath di una slice può essere semplificato, come mostrato in Figura 2.19.

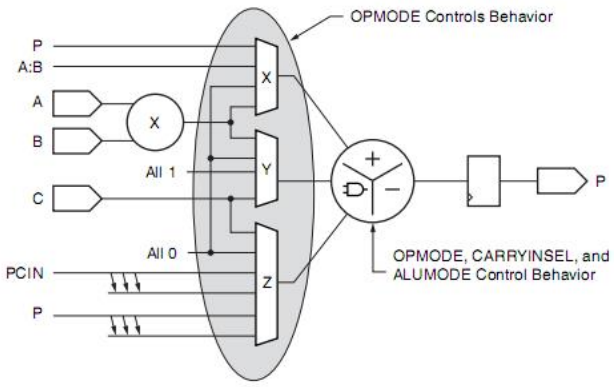

Figura 2.19 Rappresentazione semplificata delle risorse di una slice DSP48E

Gli input A e B possono attraversare una catena di due registri, in modo da favorire la realizzazione di elaborazioni differenti. La presenza dei registri favorisce la realizzazione di una struttura pipelined. Per gli altri input, sia dati che controlli, può essere inserito un solo registro di sincronizzazione, prima del loro utilizzo. La massima frequenza di funzionamento, pari a 550MHz, può essere raggiunta utilizzando i registri di pipeline.

L'Equazione 2.1 sintetizza le combinazioni di X, Y, Z e Cin realizzabile dall'unità aritmetica. I valori dell'uscita dei due multiplexer X e Y, e il Cin saranno sempre sommati insieme. Questo risultato può essere, selettivamente, sommato o sottratto all'uscita del multiplexer Z.

$$Adder/Sub\ Out = (Z \pm (X + Y + CIN)) \text{ or } (-Z + (X + Y + CIN) - 1)$$

Equazione 2.1

Attraverso il registro ALUMODE è possibile selezionare la modalità di funzionamento dell'unità logico-aritmetica. Le unità di addizionatore e sottrattore possono, inoltre, essere utilizzate in modalità SIMD. Queste unità supportano le operazioni SIMD a due vie, con operandi a 24bit, o a quattro vie a 12bit. Nel caso in cui si utilizzano le operazioni SIMD, i bit CARRYOUT vengono utilizzati come bit di riporto delle operazioni.

L'uscita del moltiplicatore, a 43 bit per non generare overflow, viene esteso in segno a 48 bit, per essere utilizzato come input dell'unità logico-aritmetica. È possibile utilizzare i 5 bit, differenza tra l'uscita del moltiplicatore e l'ingresso dell'ALU, come bit di guardia.

Nel caso in cui non viene utilizzato il moltiplicatore, possono essere implementate le funzioni logiche AND, OR, NOT, NAND, NOR, XOR, and XNOR. Gli input, per queste funzioni logiche, possono essere selezionati attraverso i multiplexer X e Z.

I dati, una volta che sono usciti dell'ALU, attraversano un pattern logico di rilevamento. Questo speciale pattern permette di implementare, nella slice, un contatore con azzeramento al raggiungimento di una soglia, senza utilizzare logica aggiuntiva. Inoltre permette di supportare il rounding convergente e l'overflow, l'underflow e la saturazione negli accumulatori. Utilizzando anche l'unità logica, il pattern logico di rilevamento può essere configurato per eseguire una comparazione dinamica a 48 bit su due operandi.

Arbitrary Waveform Generator NI PXIe-5451

L'NI PXIe-5451, rappresentato in Figura 2.20, è un generatore di forme d'onda arbitrario. Il dispositivo è dotato di due DAC a 16bit con una frequenza di campionamento di 400MS/s. i segnali d'uscita del modulo possono essere generati sia in modalità single-ended sia in modalità differenziale.

Figura 2.20 L'NI PXIe-5451, generatore di forme d'onda arbitrario utilizzato nell'EG

Ogni terminale di uscita ha un spurious-free dynamic range (SFDR) di 98dB ad 1 MHz, con il valore di -146dBc/Hz per il rumore di fase. In questo modo il PXIe-5451 diventa lo strumento ottimale per testare dispositivi con input modulati in I&Q, per generare segnali multipli a banda larga o per generare i segnali in banda base di un generatore di segnali vettoriale RF. Questo dispositivo permette, inoltre, attraverso un'unità di onboard signal processing (OSP) di elaborare i segnali direttamente nell'AWG. L'unità di OSP può eseguire operazioni di upconversion digitale, di pulse shaping, di implementare un filtro di interpolazione, di eseguire un controllo sull'offset e sul guadagno e uno spostamento in frequenza dei segnali, grazie ad un NCO, un oscillatore controllato numericamente, presente nell'AWG.

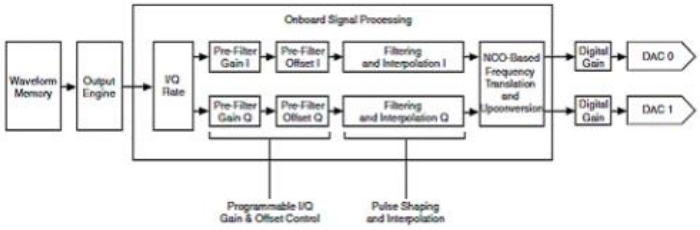

Figura 2.21 Diagramma a blocchi dell'unità di onboard signal processing (OSP) dell' L'NI PXIe-5451

Il modulo PXIe-5451 sfrutta, inoltre, le caratteristiche dal bus PXI Express per lo streaming continuo di dati. La velocità di trasferimento, dal controller host, è maggiore di 600MB/s in modalità dual-channel o di 360MB/s in modalità single-channel.

Le caratteristiche del dispositivo, utilizzate nella realizzazione dell'EG, sono:

il trasferimento di tipo Peer-to-Peer, supportato dal modulo

la possibilità di utilizzare i trigger inviati sul bus PXI Express, presente nel backplane, da altri moduli, per sincronizzare la generazione dei segnali.

Attraverso lo streaming Peer-to-Peer tra due moduli, è possibile implementare un trasferimento diretto di dati tra due unità hardware. Uno streaming Peer-to-Peer agisce come una singola e unidirezionale coda, in cui i dati possono passare da un dispositivo ad un altro. Nel caso in esame, permette di trasmettere i dati elaborati dall'FPGA direttamente all'AWG per la generazione del

segnale, senza l'intervento del sistema host e del sistema operativo. Il passaggio diretto dei dati, da FPGA all'AWG, evita l'inserimento di tempi di ritardo nella generazione dei segnali.

L'altro vantaggio è di poter utilizzare dei segnali di trigger, provenienti da altri moduli, per la temporizzazione della generazione dei segnali. In questo modo, è possibile sincronizzare un'operazione di generazione con un segnale proveniente da un dispositivo digitale, per esempio dall'FPGA.

L'interfaccia del dispositivo è mostrata in Figura 2.22. Sono presenti:

- i terminali d'uscita dei segnali, sia in configurazione single-ended che differenziale,
- il terminale di CLK IN per utilizzare un clock esterno come riferimento per la generazione,
- il terminale di CLK OUT, nel caso in cui si voglia utilizzare il clock dell'AWG in altri dispositivi
- due terminali SMB chiamati PFI0 e PFI1, per utilizzare un trigger esterno per la generazione.

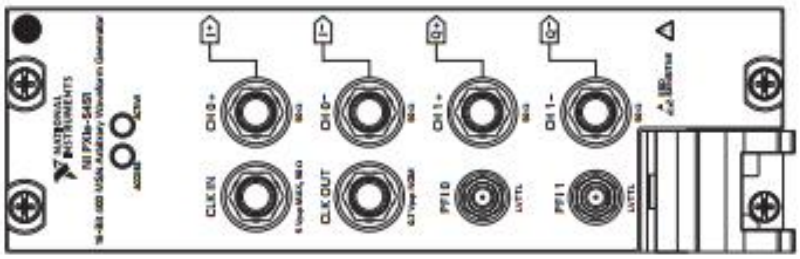

Figura 2.22 Interfaccia esterna dell'NI PXIe-5451

Modulo Adapter NI 5781

Il dispositivo NI 5781 è un modulo adapter dual-input e dual-output. È predisposto per interfacciarsi con segnali in banda base e, eventualmente, con mixer che permettono la traslazione del segnale nella banda reale. Il modulo adapter forma, con il modulo FlexRIO per la programmazione di FPGA, un ricetrasmettitore in banda base. Questo ricetrasmettitore può essere utilizzato per realizzare una modulazione e demodulazione RF, per emulare un canale di trasmissione, o per l'analisi spettrale di un segnale. Inoltre, per la presenza dell'FPGA, consente un controllo rapido delle applicazioni la possibilità di un'elaborazione in-line dei dati. Questi risultati possono essere raggiunti, grazie ad un basso tempo di risposta e un'alta velocità di elaborazione dell'FPGA.

Il dispositivo ha un ADC a 14bit per campionare, anche contemporaneamente, due ingressi analogici differenziali o single-ended, a 100MS/s. È presente un DAC a 16bit per la generazione, anche simultanea, di due uscite analogiche differenziali o single-ended, a 100MS/s. L'NI 5781 dispone, inoltre, di altri tre connettori: due MCX, per il collegamento del clock input e del clock output e l'altro per l'utilizzo degli I/O digitali attraverso un connettore a 9 pin.

Il modulo, per gli ingressi analogici, presenta un filtro con frequenza di taglio a 40MHz, per non violare il teorema di Nyquist in fase di acquisizione e generazione di un segnale. Il teorema di Nyquist afferma che, in una conversione analogico-digitale, la minima frequenza di campionamento

necessaria per evitare ambiguità e perdita di informazione, nella ricostruzione del segnale analogico originario (ovvero nella riconversione digitale-analogica), con larghezza di banda finita e nota, è pari al doppio della frequenza del segnale stesso. I filtri passabasso con frequenza di taglio di 40MHz, mostrati nel diagramma a blocchi in Figura 2.23, bloccano tutte le componenti frequenziali al di sopra dei 50 MHz, limite imposto dal teorema di Nyquist.

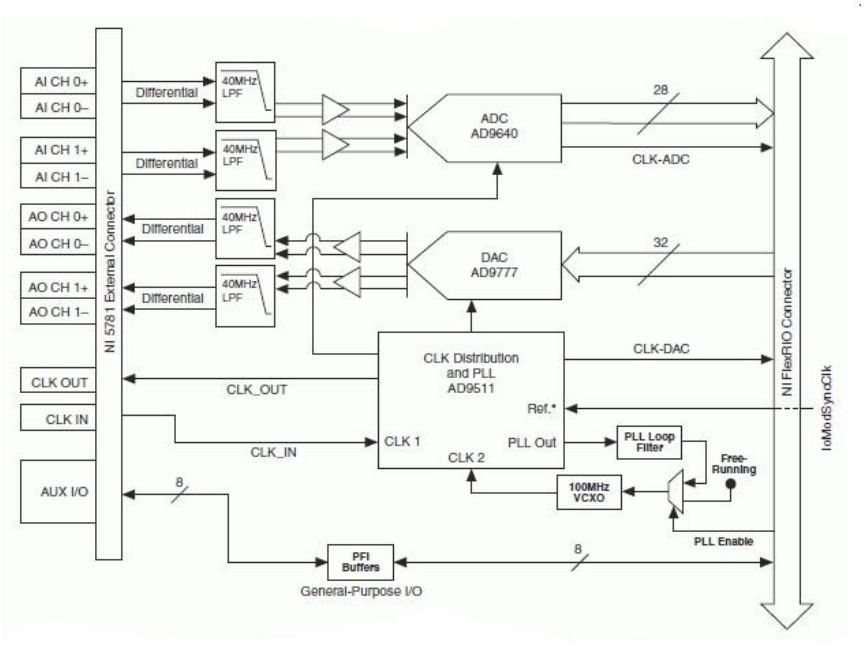

Figura 2.23 Schema a blocchi del modulo I/O Adapter NI 5781

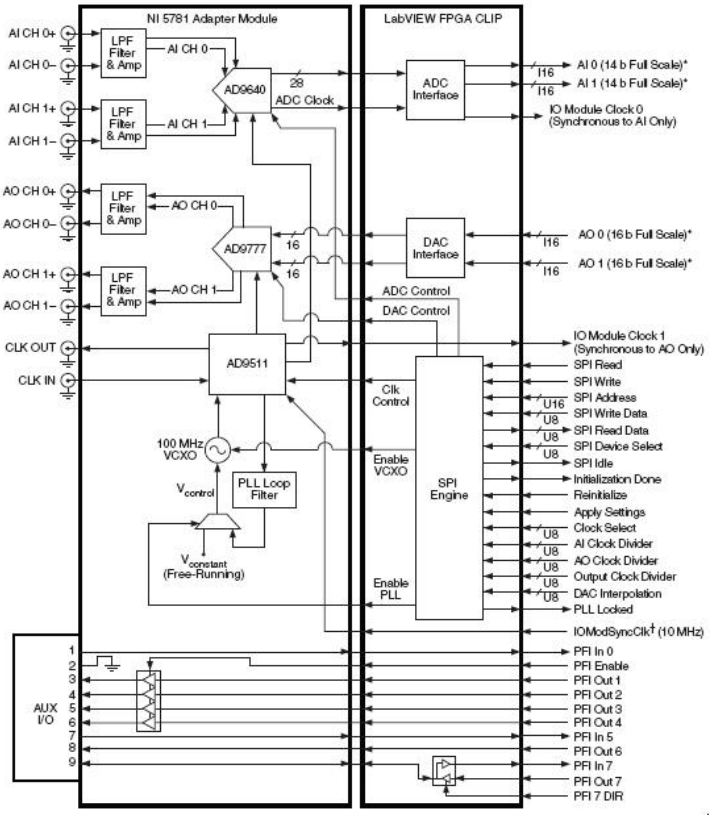

Figura 2.24 Connettori e diagramma a blocchi dei segnali per il NI 5781

Lo schema a blocchi del modulo adapter è rappresentato in Figura 2.23. In questa figura, vengono messi in evidenza i filtri presenti nelle porte analogiche; i convertitori AD9640 e AD9777, utilizzati per la conversione analogico-digitale e digitale-analogico dei segnali; il circuito di distribuzione del clock e il PLL, utilizzabile per l'eventuale gestione del clock esterno. Laddove non fosse presente il clock esterno, si può utilizzare, per il campionamento e la generazione dei segnali, un clock di 100MHz, generato del VCO interno come frequenza di free running del VCO.

In Figura 2.24, invece, vengono mostrati alcuni buffer per l'abilitazione dei terminali di I/O digitali. Il clock, utilizzato all'interno del modulo adapter, può essere:

- Il clock a 40MHz, proveniente dall'FPGA,
- Il clock a 100MHz, generato dal VCO in free running,
- Il clock esterno, da collegare alla porta Clk IN del modulo adapter.

Il clock esterno, per essere utilizzato correttamente all'interno del modulo e anche nell'FPGA, deve essere caratterizzato da una frequenza, compresa tra 20MHz e 100MHz; da un'ampiezza, compresa tra 400mV_{pk-pk} e 2 V_{pk-pk} e da un duty cycle, compreso tra 40% e il 60%.

Modulo Adapter NI 5761

Il dispositivo NI 5761 è un adapter module per NI FlexRIO. Fornisce all'FPGA presente nella FlexRIO, un'interfaccia analogica per l'acquisizione di segnali ad alta frequenza. Il adapter module è dotato di due ADC a 14bit. Con una larghezza di banda di 500MHz e una frequenza di campionamento di 250MS/s, l'NI5781 può essere utilizzato per un sotto campionamento dei segnali alle frequenze intermedie IF, usate nelle telecomunicazioni, fino alla quarta regione di Nyquest. Inoltre fornisce 8 linee digitali I/O per personalizzare le operazioni di trigger o di comunicazione con altri dispositivi. Questo modulo utilizza il chip ADS62P49 della Texas Instruments, un ADC con una risoluzione di 14bit e con caratteristiche dinamiche eccellenti.

Nella Figura 2.25 viene mostrato il diagramma a blocchi dell'NI 5761.

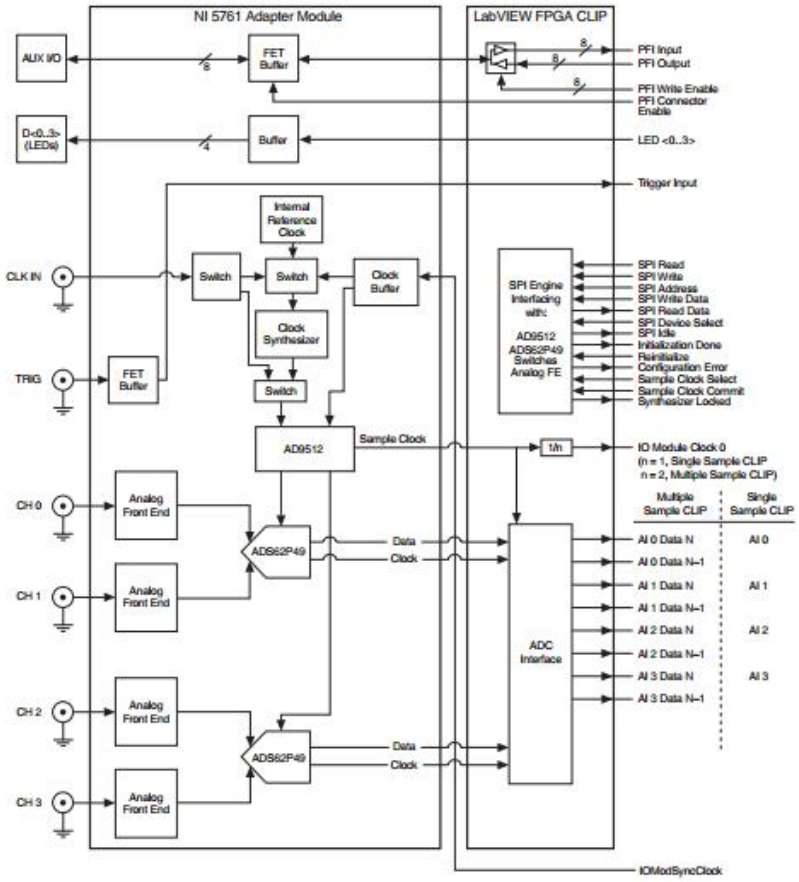

Figura 2.25 Diagramma a Blocchi dell'NI 5761

Nella parte inferiore del diagramma si possono osservare gli ADC. Ognuno è collegato ad un front end analogico, prima di interfacciarsi con i segnali collegati all'adapter module. I dati provenienti dagli ADC arrivano ad un'interfaccia, chiamata ADC Interface, che ne permette una corretta interpretazione. Sono presenti sia una linea di clock che una linea dati, tra gli ADC e l'interfaccia, perché viene utilizzato un protocollo seriale per la trasmissione dei dati. Il modulo Labview FPGA include degli IP in HDL, per facilitare l'implementazione della comunicazione tra l'FPGA e l'adapter Module, chiamati CLIP. Questo modulo permette di utilizzare due tipi di CLIP:

- CLIP personalizzate dall'utente
- CLIP socketed (che permettono all'utente di comunicare direttamente con l'adapter module all'interno di un VI FPGA).

Le clip socketed sono delle CLIP standard, messe a disposizione dall'ambiente di sviluppo, per semplificare utilizzo dell'adapter module. Sono presenti tre tipi di CLIP: Multiple Sample CLIP, Single Sample CLIP e Low Speed CLIP. La Multiple Sample CLIP permette di acquisire due campioni per ogni ciclo di clock. Questa frequenza, utilizzata sull'FPGA, risulta essere la metà della reale frequenza di campionamento. Nella Figura 2.26, viene mostrata l'acquisizione di due campioni, *N* e *N-1*, per ciclo di clock.

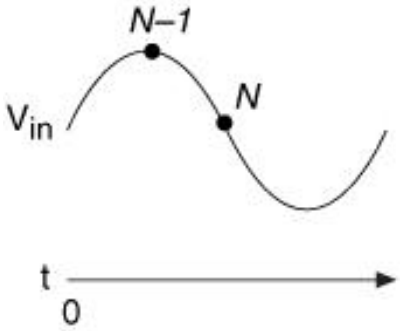

Figura 2.26 Rappresentazione temporale dell'acquisizione con la Multiple Sample CLIP

Per esempio, se si vuole utilizzare una frequenza di campionamento di 250MHz, è possibile eseguire un codice su FPGA a 125MHz. Questa soluzione è realizzabile perché la CLIP permette di avere a disposizione i due campioni, N e N-1, nella stessa iterazione di un ciclo, permettendo di eseguire il codice su FPGA ad una frequenza minore, con maggiore facilità nella realizzazione del codice.

La Simple Sample CLIP, diversamente dalla Multiple Sample CLIP, genera un campione per ciclo di clock. Può essere utilizzato il clock interno, predefinito, di 250MHz o un clock esterno ad una frequenza differente. In questo caso, il codice FPGA che utilizza i dati dell'ADC, dovrà funzionare a 250MHz. Quindi la scrittura del codice diventa più complessa.

La Low Speed CLIP, genera un campione per ciclo di clock, ma con frequenze di campionamento minori o uguali a 150MHz. In questo caso è possibile utilizzare solo un clock esterno.

Nella parte superiore del diagramma a blocchi in Figura 2.25, vengono mostrate le linee digitali, bufferizzate e indicate come PFI; gli ingressi per il clock esterno e per un trigger esterno. Sono inoltre presenti nell'interfaccia verso l'FPGA, delle linee digitali SPI per settare alcuni parametri nei registri presenti nell'ADC.

Configurazione completa dell'EG

La configurazione hardware finale dell'EG è basata su di un sistema PXI. All'interno dello chassis PXIe-1082 sono presenti:

- un controller PXIe-8133
- un modulo PXIe-7965R con un adapter module NI 5761
- un modulo PXIe-7965R con un adapter module NI 5781
- un modulo NI PXIe-5451

Una rappresentazione del sistema definitivo, senza gli adapter module, è mostrato nella Figura 2.27.

Figura 2.27 Rappresentazione dell'EG, senza gli adapter module

Una foto del sistema è mostrata in Figura 2.28.

Figura 2.28 Foto del sistema completo, in cui sarà implementato l'EG.

Nel sistema, mostrato in Figura 2.28, è presente anche il modulo NI 8260, un controller di dischi SSD ad alta velocità. Questo modulo, però, non è utilizzato nella realizzazione dell'EG in questa configurazione, ma in una configurazione definita "off-line". Non è presente il modulo che permette la comunicazione con la reflective memory perché ancora non è stata consegnata al consorzio CO.RI.S.T.A. l'unità per simulare gli input. Quindi gli input, provenienti dalla reflective memory, saranno simulati direttamente su FPGA.

Tecniche di programmazione per Labview FPGA

LabVIEW è un ambiente di sviluppo per il linguaggio di programmazione visuale che permette una programmazione di alto livello. LabVIEW viene utilizzato per acquisizione, analisi ed elaborazione digitale dei dati, controllo di processi, o più generalmente per tutto ciò che concerne i sistemi di misura e l'automazione industriale. Il linguaggio di programmazione usato in LabVIEW si distingue dai linguaggi tradizionali perché grafico, e non basato su un'interfaccia di tipo testuale con righe di comando. La definizione di strutture dati ed algoritmi avviene con icone e altri oggetti grafici, ognuno dei quali incapsula funzioni diverse, uniti da linee di collegamento (wire), in modo da formare una sorta di diagramma di flusso. Tale linguaggio viene definito dataflow (flusso di dati) in quanto la sequenza di esecuzione è definita e rappresentata dal flusso dei dati stessi, attraverso i fili monodirezionali che collegano i blocchi funzionali. L'esecuzione di un blocco è possibile solo se sono disponibili tutti i suoi input. Poiché i dati possono anche scorrere in parallelo, attraverso blocchi e fili non consecutivi, il linguaggio realizza spontaneamente il multithreading, senza bisogno di esplicita gestione. Un programma o sottoprogramma, denominato VI, non esiste sotto forma di testo, ma può essere salvato solo come un file binario, visualizzabile e compilabile solo da LabVIEW. Un VI è composto, com'è mostrato in Figura 2.29, da due elementi principali: il Front Panel ed il Block Diagram. Il Front Panel fornisce un'interfaccia utente del programma, mentre il Block Diagram permette di visualizzare il codice grafico.

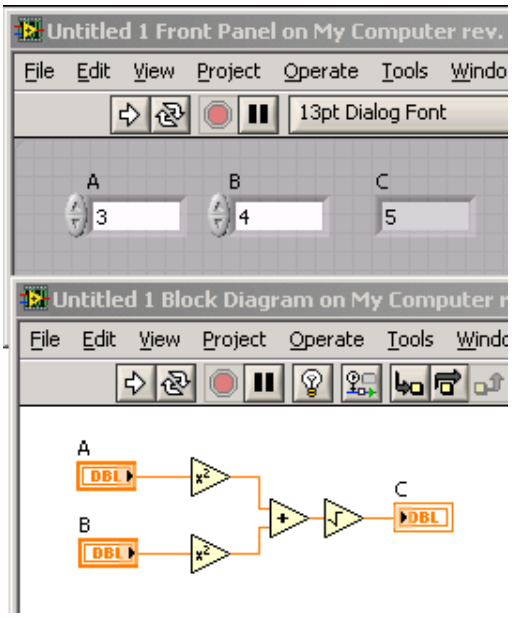

Figura 2.29 Front Panel e Block Diagram di un VI Labview

Il Front Panel è l'interfaccia utente del VI e viene realizzato attraverso controlli e indicatori, che costituiscono i terminali interattivi, rispettivamente, d'ingresso e d'uscita. Il Block Diagram è il diagramma di flusso che rappresenta il codice sorgente in formato grafico. Nel Block Diagram possono essere eseguite delle chiamate ad altri VI, che vengono chiamati SubVI.

LabVIEW FPGA Module, modulo aggiuntivo di LabVIEW, permette di estendere le funzionalità dell'ambiente di sviluppo grafico su target FPGA (Field-Programmable Gate Arrays). LabVIEW FPGA permette una gestione integrata, per il trasferimento dati, dai moduli I/O al chip FPGA e dal

chip FPGA al processore. Inoltre permette l'analisi e l'elaborazione dei dati, la temporizzazione e la sincronizzazione integrata dei programmi, l'utilizzo di I/O analogici e I/O digitali, la realizzazione di contatori e protocolli di comunicazione digitale personalizzabili.

Nella configurazione hardware utilizzata per l'EG, l'FPGA è connessa al processore, presente nel controller, attraverso un bus PCI ad alta velocità. L'FPGA, installata nel modulo FlexRIO, può essere programmata e consente la comunicazione con i moduli adapter installati. La presenza dell'FPGA permette dei tempi di elaborazione ridotti, con un aumento delle prestazioni del sistema in termini di velocità.

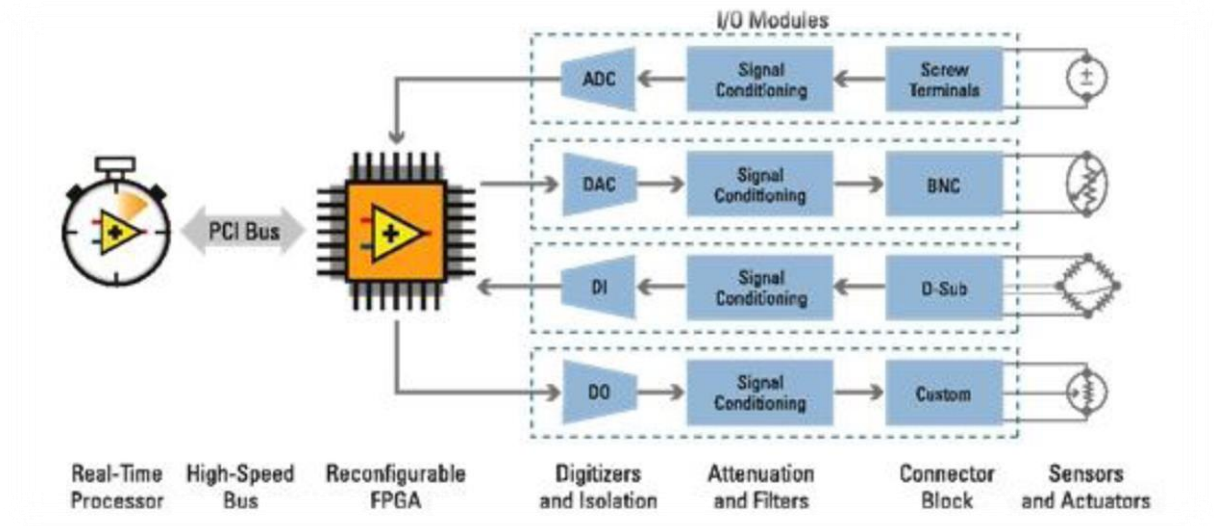

Figura 2.30 Schema della struttura CompactRIO

Si possono individuare due elementi fondamentali nella programmazione utilizzando Labview FPGA:

- **Processore presente nel controller, denominato Host**: è il PC con il quale si programma l'FPGA. Sul processore possono essere implementate delle fasi di elaborazione e analisi dei dati più complesse (per esempio, un'interfaccia esterna verso la quale l'FPGA può inviare i dati acquisiti, per essere presentati e con la quale può comunicare per ricevere segnali di input e comandi);

- **FPGA**: il dispositivo digitale programmabile attraverso cui comunicare con i moduli I/O installati, e su cui implementare le operazioni ad alta priorità.

Attraverso il linguaggio grafico, diventa più semplice sfruttare il parallelismo tra i processi eseguiti su FPGA, grazie alla possibilità di seguire il flusso di dati. L'utilizzo del modulo Labview FPGA permette di avere, nella programmazione dell'FPGA:

- Flussi multipli, su FPGA, di dati da più ingressi analogici
- Intere parti di codice eseguite in un unico colpo di clock, con la possibilità di selezionare il clock da utilizzare
- Processing di dati parallelo, in modo da aumentare la capacità computazionale, senza ridurre la velocità dell'applicazione FPGA

Un esempio di questa logica è illustrato in Figura 2.31, in cui le funzioni F=(A+B)C e Z=X+Y+M sono implementate parallelamente in LUT separate, per migliorare delle prestazioni.

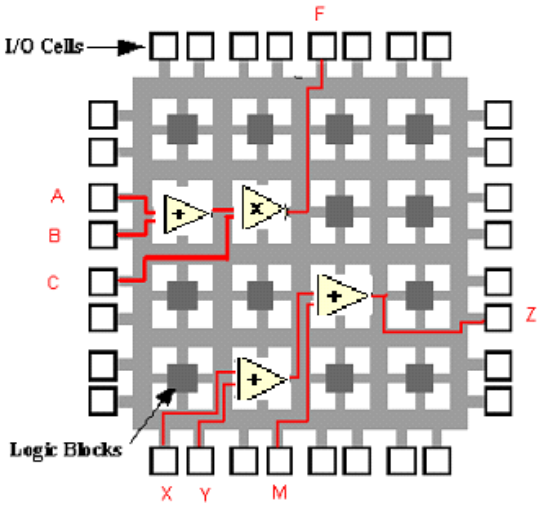

Figura 2.31 Realizzazione circuitale per l'esecuzione
parallela su FPGA di due funzioni

Un sistema implementato su FPGA permette di implementare una strategia di controllo dei programmi differente. Le scelte non vengono prese a livello dell'applicazione software, ma arrivano a livello hardware, direttamente nell'FPGA, permettendo una maggiore velocità e una maggiore resistenza ai crash del sistema.

Come si può vedere dalla Figura 2.32, nel caso di un sistema con FPGA, la posizione in cui vengono eseguite le operazioni è vicino all'interfaccia I/O. Nel caso di un sistema tradizionale, i controlli sono nel livello delle applicazioni software con una maggiore probabilità di crash del sistema.

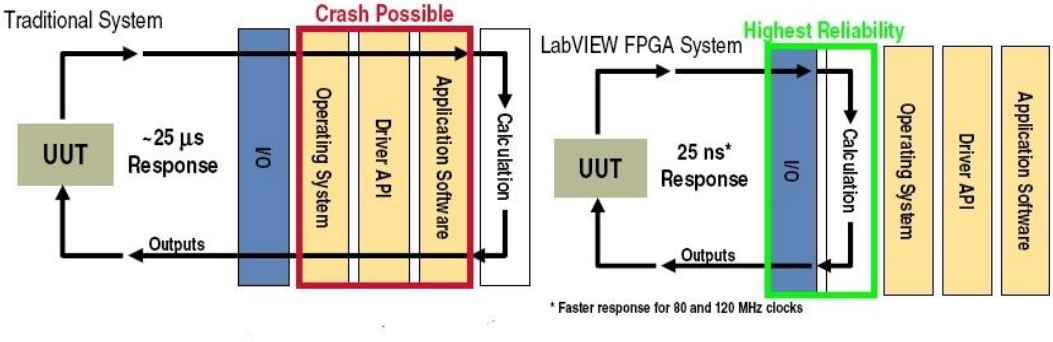

Figura 2.32 A sinistra un sistema tradizionale con le elaborazioni eseguite al livello più alto.
A destra un sistema con l'utilizzo di FPGA ed elaborazioni a livello dell'IO

Nei VI sviluppati su FPGA spesso vengono eseguiti i controlli ad alta priorità, le acquisizioni, la temporizzazione, trigger ed elaborazione In-line (un'elaborazione integrata che si può eseguire senza l'accumulo di molti dati). Le operazioni di I/O da file, l'interfaccia utente, il controllo per la segnalazione degli errori ed l'elaborazione di dati con alto impiego di tempo e di dati sono eseguiti dal processore. In questo caso, è necessario eseguire queste operazioni sul processore perché si ha bisogno di una quantità di memoria elevata per l'elaborazione di una grande quantità di dati.

Per creare un programma che potrà essere eseguito su FPGA, si deve aggiungere un VI all'interno della finestra per l'esplorazione del progetto, sotto il target che si vuole programmare. Per la creazione del VI, si deve premere il pulsante destro sul modulo che contiene l'FPGA, e selezionare **New**, **VI** e viene aggiunto un VI FPGA al progetto. Il Front Panel di un VI, da eseguire su FPGA, utilizza dei normali controlli e indicatori, che vengono impiegati come elementi per lo scambio di dati. Quindi, il Front Panel è solo l'interfaccia che si usa con il processore.

Il Block Diagram mostra il vero e proprio codice grafico per la programmazione dell'FPGA.

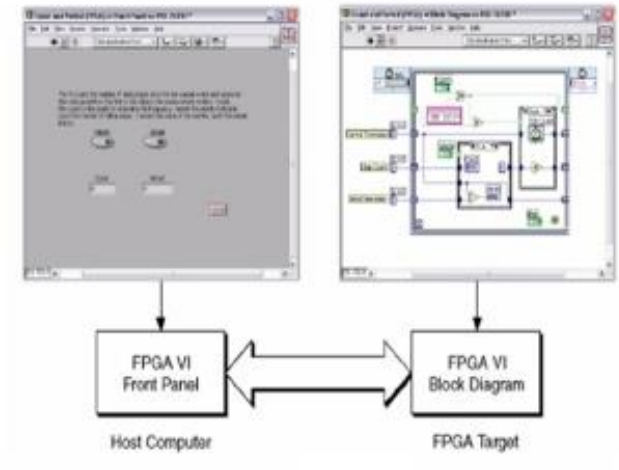

Figura 2.33 Comunicazione attraverso il Front Panel interattivo

Le palette, in cui sono contenuti gli oggetti grafici per la programmazione di un VI FPGA, sono molto simili a quelle presenti nel Labview per il sistema Host. Sono solo aggiunte delle voci specifiche per l'FPGA come FPGA I/O, Memory & FIFO e FPGA Math & Analysis. Queste modifiche sono necessarie perché mostrano delle funzioni specifiche per il target e necessarie solo su FPGA, come per esempio per la gestione di una FIFO.

La programmazione di un dispositivo FPGA avviene attraverso la configurazione di switch d'interconnessione, di Look-up Table (LUT) e d'interconnessioni tra i vari CLB con l'utilizzo di un file Bitstream. La generazione di questo file avviene nella fase di compilazione con la conversione del VI FPGA in una serie di bit adatta per la configurazione dell'FPGA.

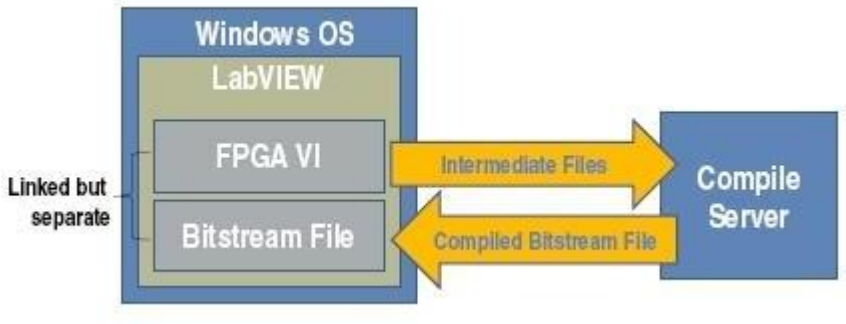

Figura 2.34 Passi per la compilazione di un VI FPGA

Il modulo Labview FPGA esegue una prima sintesi del VI, descritto con un codice grafico, in linguaggio VHDL, generando dei file intermedi. In questa fase si esegue l'analisi delle gerarchie (se vi sono dei SubVI nel programma che compilo), l'analisi dei diagrammi e la creazione dei file intermedi.

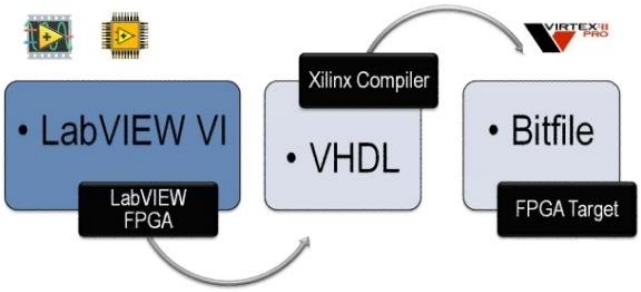

Figura 2.35 Compilazione di un programma con Labview FPGA

I file intermedi (in cui il programma è descritto con il codice VHDL) vengono utilizzati nel server della compilazione, lo Xilinx ISE Compiler, per effettuare il passaggio dal linguaggio VHDL alla creazione del bitfile. In questa fase si esegue la generazione di altri file intermedi, sempre con linguaggio HDL, e viene effettuata la compilazione, l'analisi delle gerarchie, l'analisi e la verifica delle entità e la sintesi del programma. Dopo l'analisi, in cui il compilatore conosce quali strutture compongono il programma, viene effettuata una sintesi avanzata del codice HDL in cui, per esempio, si ottimizza la codifica delle macchine a stati finiti. Un'ultima analisi di basso livello permette di costruire la versione finale della Netlist per la descrizione del circuito. La successiva fase di Mapping realizza il primo posizionamento delle LUT su FPGA, definisce le regioni di clock e il routing dei segnali. Il successivo Place And Route migliora e ottimizza la logica utilizzata. Infine viene generato il file Bitstream che servirà per la programmazione del target FPGA.

Una volta completata la compilazione, la finestra Compilation Status visualizza dei report sulla temporizzazione e sui dispositivi utilizzati nella fase di sintesi e di mapping, e lo Xilinx Log, in cui sono riportati i dettagli della compilazione. Una volta chiusa questa finestra si può eseguire il VI sull'FPGA.

FPGA I/O

Gli funzioni FPGA I/O permettono la connessione tra i terminali I/O e la logica FPGA. Ogni elemento ha un nome e un tipo specifico, per distinguere tra i vari terminali di I/O analogici e digitali. Con terminale s'indica la connessione hardware tra il modulo FlexRIO e l'interconnessione verso l'esterno. Con risorsa I/O s'indica la rappresentazione in Labview FPGA di un terminale.

Per utilizzare un terminale I/O si ha bisogno di inserire un I/O node, individuabile nelle palette di programmazione (selezionando FPGA I/O nella categoria programming), all'interno del VI. Prima di usufruire attivamente di un I/O node, bisogna selezionare quale terminale rappresenta il nodo I/O, come indicato in Figura 2.36. Ci sono due modi per utilizzare i nodi I/O. Si esegue un drag and drop (trascinare l'elemento) dalla finestra per l'esplorazione del progetto all'interno del programma, o si aggiungere un nodo I/O vuoto nel Block Diagram. In questo caso, cliccando su di esso con il pulsante sinistro del mouse, si può selezionare la risorsa I/O che si vuole utilizzare.

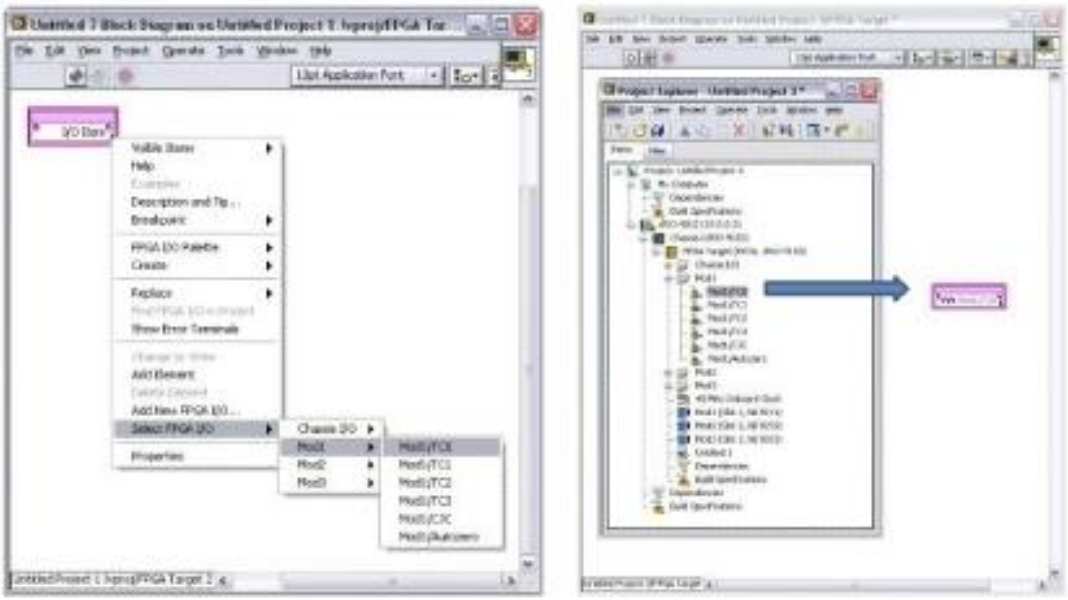

Figura 2.36 Modalità di utilizzo dei nodi I/O

Sono presenti vari tipi di I/O, come i nodi I/O per i terminali digitali, in cui si può scrivere o leggere dei valori booleani, e i nodi I/O per le porte analogiche, in cui si può leggere o scrivere un valore intero. Con il modulo Adapter NI 5781, si hanno a disposizione 2 canali per l'input analogo, due per l'output in cui è possibile scrivere dei valori interi a 16bit e 7 linee digitali. Invece il modulo Adapter NI 5761, mette a disposizione 4 canali per l'input analogo, in cui è possibile leggere dei valori interi a 16bit e 8 linee digitali.

Temporizzazione di un VI FPGA

Per una temporizzazione rapida dei VI, vengono utilizzate alcune funzioni che permettono il controllo della velocità di ripetizione delle strutture (che indicano, in Labview, i cicli e gli elementi di selezione).

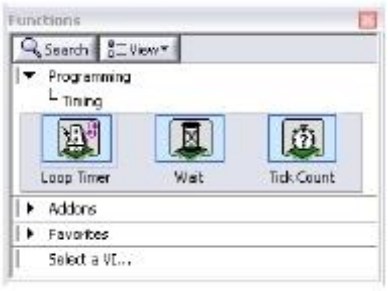

Figura 2.37 Funzioni di temporizzazione

La prima funzione utilizzabile è quella di **Wait,** con cui si specifica un tempo minimo, dichiarato dal programmatore, di esecuzione per una sequenza di operazioni. Se le istruzioni vengono concluse in un tempo più breve di quello indicato tramite la suddetta funzione, allora questa bloccherà l'esecuzione fin che non sarà trascorso il tempo minimo di esecuzione.

I contatori vengono implementati direttamente in Hardware. Possono utilizzare diverse dimensioni e diverse unità per effettuare il conteggio. Questo permette di realizzare un controllo su di un intervallo molto ampio, rispetto alle operazioni normalmente eseguite su FPGA. La dimensione interna del contatore determina il tempo massimo che può misurare, ma con una dimensione maggiore si utilizza più logica per la sua realizzazione. Per risparmiare logica riconfigurabile, quindi, conviene utilizzare sempre la minima dimensione possibile. La dimensione Tick indica il periodo del clock utilizzato e, se viene utilizzato il clock on-board dell'FPGA di 40 MHz, il periodo corrispondente sarà di 25ns (è utile se si deve effettuare un timing dell'ordine dei nanosecondi).

Size of Internal Counter	Ticks	µs	ms
32-bit	1 to 4,294,967,296 ticks	1 µs to 71 minutes	1 ms to 49 days
16-bit	1 to 65,536 ticks	1 µs to 65 ms	1 ms to 65 seconds
8-bit	1 to 256 ticks	1 µs to 256 us	1 ms to 256 ms

Figura 2.38 Dimensione dei contatori e tempo massimo di conteggio
in base alle unità di misura utilizzate

La funzione **Tick Count** permette di visualizzare il valore del contatore del clock sull'FPGA. Questa funzione può essere utilizzata per testare la velocità di ripetizione dei cicli, o per misurare il tempo intercorso tra due eventi.

Un'altra funzione utilizzata per la temporizzazione è il **Loop Timers** in cui si definisce il tempo di esecuzione minimo per ogni ciclo. La differenza, con la funzione Wait, è che il Loop Timer non viene eseguito la prima volta che viene richiamata. In questo modo, il primo ciclo non viene temporizzato ma solo i successivi, il Wait invece viene eseguito ad ogni iterazione del ciclo.

Con queste funzioni è possibile implementare dei cicli di esecuzione temporizzati, utilizzando le strutture While e For. Con il ciclo **While,** è possibile costruire una struttura in cui le operazioni vengono eseguite indefinitamente e senza interruzioni. Questa struttura è importante per il codice da eseguire indefinitamente su FPGA. Può essere realizzata collegando una costante booleana falsa al controllo del ciclo While che, testando la condizione, non troverà mai l'evento per terminare il ciclo.

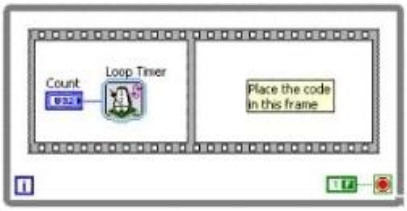

Figura 2.39 Ciclo While temporizzato ed eseguito continuamente

La programmazione con l'utilizzo di più cicli consente un'**esecuzione parallela** e con diverse frequenze di ripetizione. Questo permette di eseguire due operazioni con velocita differenti, senza

dover rallentare le operazioni che potrebbero essere eseguite più velocemente. Nell'esempio mostrato in Figura 2.40, si impiegano 35tick per utilizzare un output analogico, 1tick per utilizzare un input digitale ed altri 2 tick per il controllo della condizione del while.

Se vengono utilizzati in un unico ciclo di esecuzione, l'input digitale sarà limitato dalla frequenza di ripetizione dell'output analogico. In questo modo si devono attendere 38tick per una nuova esecuzione, sia per l'output analogico, sia per l'input digitale.

Invece se si eseguono i due cicli separatamente, si ha la possibilità di eseguire ogni parte con la propria velocità massima e senza rallentamenti. Descrivendo il codice come mostrato nella parte destra della Figura 2.40, si avrà che il ciclo con il solo output analogico sarà eseguito in 38tick e il ciclo con l'input digitale sarà eseguito in 4tick.

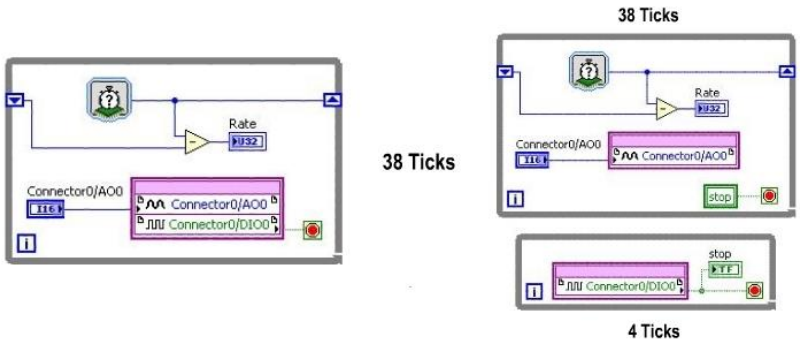

Figura 2.40 Differente temporizzazione in base
alla distribuzione del codice eseguito

Nella programmazione di una FGPA, si deve tener conto di rispettare un certo **dataflow,** necessario per la sincronizzazione dei dati con l'utilizzo di Flip Flop. Così una semplice operazione come andare a scrivere un valore in un'uscita digitale proveniente da un controllo booleano e invertito, in un linguaggio di alto livello, come il linguaggio grafico, viene realizzata velocemente.

Si può pensare che l'esecuzione si possa compiere in un unico colpo di clock, ma se si guarda l'implementazione circuitale, si capisce che si ha bisogno di più cicli per completare questa operazione.

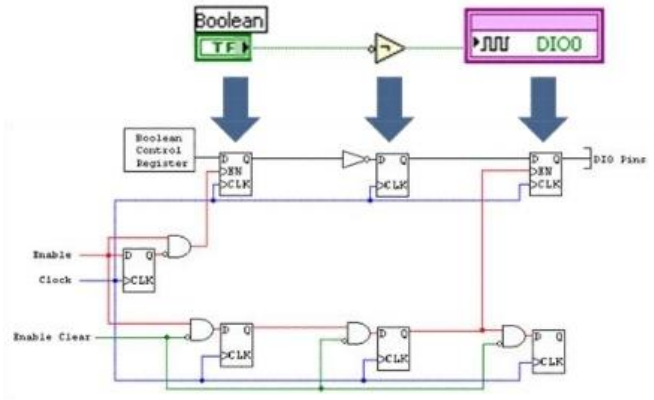

Figura 2.41 Realizzazione schematica di codice
Labview di alto livello

Ogni funzione in un VI, per essere eseguita, ha bisogno di un minimo di un periodo di clock. Se, poi, alcune funzioni devono essere eseguite in maniera sequenziale, per una dipendenza tra i dati, si ha bisogno di più cicli di clock. La frequenza di esecuzione di un ciclo while dipenderà dal tempo di esecuzione del codice all'interno della struttura, con l'aggiunta ad un overhead di due colpi di clock.

Così un ciclo può essere eseguito in un numero variabile di tick, come il codice in Figura 2.42, in cui s'impiegano dodici periodi di clock per ogni iterazione.

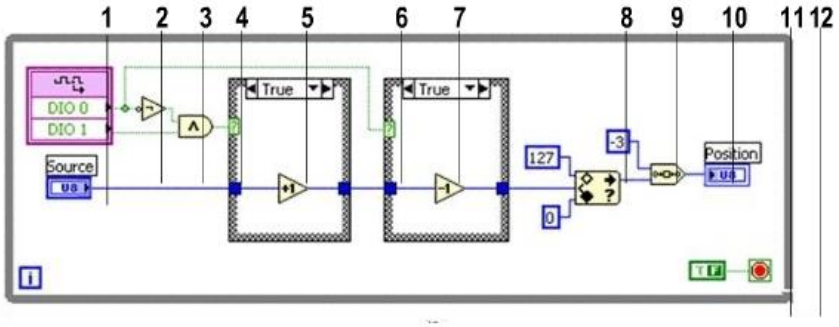

Figura 2.42 Realizzazione di un contatore in funzione degli ingressi digitali e relativa temporizzazione (con i numeri interni s'indicano i periodi di clock necessari per ogni parte del codice)

Utilizzando la struttura **Single-Cycle Timed Loop,** è possibile eseguire un'iterazione del ciclo while in un unico periodo di clock. Così si può avere una frequenza di ripetizione fissa, non dipendente dal codice da eseguire all'interno del ciclo. Lo stesso codice, implementato in Figura 2.42, utilizzando questa struttura può essere eseguito in un unico colpo di clock. In questo modo viene agevolata la programmazione per eseguire, per esempio, la temporizzazione di un codice. La scelta del clock deve essere eseguita per ogni Single-Cycle Timed Loop, in alto a sinistra del ciclo stesso, selezionando uno tra i clock disponibili e abilitati sull'FPGA. Ogni ciclo sarà eseguito in un periodo del clock, calcolabile come l'inverso della frequenza del clock. Se viene utilizzato il clock di default di 40MHz, generato all'interno del target, un tick indicherà un periodo di 25ns.

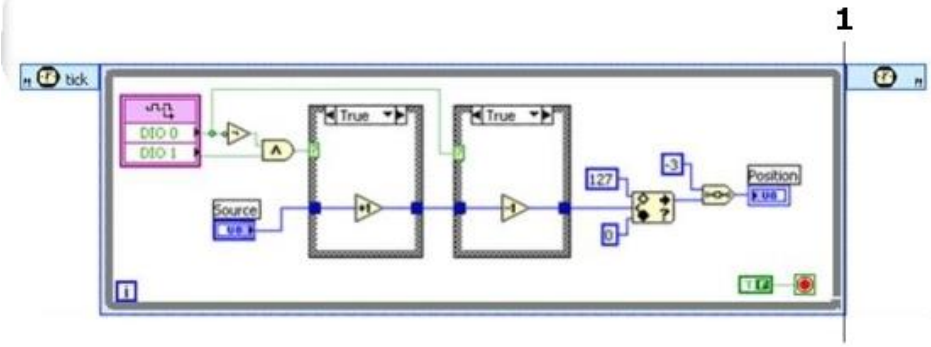

Figura 2.43 Utilizzo di un Single-Cycle Timed Loop ed esecuzione in un unico periodo di clock

Labview automaticamente ottimizza il codice all'interno del ciclo, eliminando registri sequenziali, per ottenere un'esecuzione rapida del codice. In questo modo, tutto il codice presente in un Single-

Cycle Timed Loop sarà eseguito in un unico colpo di clock. Il Single-Cycle Timed Loop può essere inserito, all'interno di un VI, selezionando nelle palette Strutture, Timed Structure e Timed Loop. Premendo il pulsante sinistro del mouse sul VI, si può creare un rettangolo, all'interno del quale è possibile inserire il codice da eseguire in un unico ciclo di clock. In questa struttura non è possibile inserire alcune funzioni e VI, come ad esempio altri Single-Cycle Timed Loop, dei Loop Timer, la richiesta di un interrupt e delle funzioni Wait. Queste funzioni richiedono, per la loro esecuzione, più di un periodo di clock. Se ci sono degli oggetti non supportati all'interno della struttura, nella compilazione viene segnalato un errore all'utente nella generazione dei file intermedi, senza neanche iniziare la compilazione con il componente Xilinx.

Scambio di dati in un VI FPGA

Eseguendo le elaborazioni di dati parallelamente, si ha la necessità di poter condividere i dati tra due cicli, ossia di poter effettuare un passaggio di dati da un ciclo ad un altro.

Questa funzionalità non può essere implementata con un collegamento diretto wire. Il passaggio dei dati, con solo un collegamento, avverrebbe solo alla conclusione del ciclo che invia i dati. Inoltre il loop che deve ricevere le informazioni, non può iniziare la sua esecuzione fin quando non ha a disposizione tutti i dati d'input. Per questo sono necessarie delle risorse che possono essere condivise tra due cicli come la memoria, le FIFO e le variabili locali.

Le variabili locali sono degli elementi del Block Diagram che permettono l'accesso simultaneo, da parte di due cicli, di una risorsa condivisa. Per esempio, un indicatore può essere utilizzato per eseguire l'abilitazione di due cicli contemporaneamente. Vi sono due tipi di variabili:

- la variabile locale, accessibile solo dal singolo VI in cui è creato;
- la variabile globale accessibile da ogni VI eseguito sul target FPGA.

È possibile creare una variabile globale premendo il pulsante destro del mouse, sul controllo o sull'indicatore del Block Diagram cui si vuole far riferimento, e selezionando Create e poi Local Variable.

Un esempio di una variabile locale, per la condivisione di un dato tra due cicli, è mostrato in Figura 2.44.

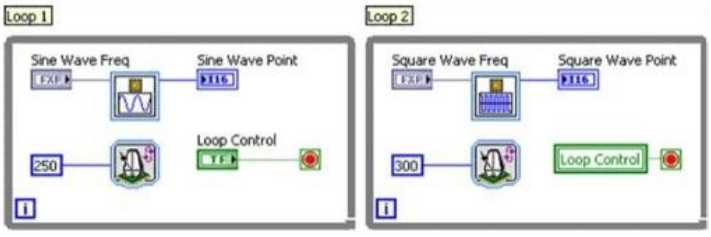

Figura 2.44 Utilizzo delle variabili locali per il controllo
di due cicli per la generazione di sinusoidi

Gli **elementi di memoria** vengono adoperati per risolvere il problema della condivisione dei dati tra più VI eseguiti su FPGA. Sono implementati in maniera molto simile alle variabili locali ma, in

questo caso i valori più recenti, che assume la variabile, vengono memorizzati. La creazione di elementi di memoria può essere fatta premendo il pulsante destro del mouse, sul target presente nella finestra di esplorazione del progetto, selezionando New e poi Memory.

I dati possono essere salvati utilizzando la memoria RAM, presente direttamente sull'FPGA, o si possono utilizzare delle LUT, rinunciando a delle risorse logiche. Si utilizzano i blocchi di memoria quando si ha bisogno di una grande quantità di memoria. L'utilizzo delle LUT viene indicato quando si ha bisogno di effettuare, gli accessi in memoria, in un unico colpo di clock, oppure si ha la necessità di avere un'alta velocità di lettura dei dati. Un'altra situazione vantaggiosa per l'utilizzo delle LUT può essere il caso in cui, l'ammontare della memoria di cui si ha bisogno, è inferiore rispetto alla minima grandezza dei blocchi di RAM o se non si hanno abbastanza blocchi di RAM liberi sull'FPGA.

I blocchi di memoria possono essere configurati in due modalità

- VI-Defined in cui la memoria si configura per l'accesso da un singolo VI
- Target-Scoped per l'accesso in tutti i programmi implementati nel target

Le **FIFO** sono un altro strumento per il trasferimento dei dati. La struttura FIFO è utilizzata come un buffer, con dimensione fissa, in cui il primo dato scritto sarà il primo letto e cancellato dalla memoria. La creazione di una FIFO è molto simile alla creazione degli elementi di memoria. Può essere fatta premendo il pulsante destro del mouse sul target, presente nella finestra di esplorazione del progetto, selezionando New e poi FIFO. Questa procedura apre una finestra in cui si può configurare la modalità, il tipo di dati, la dimensione, la sua implementazione ed altre configurazioni avanzate della FIFO. Si deve definire il numero di elementi, che la FIFO può memorizzare, poiché si dispone di una memoria limitata sul dispositivo. Il numero massimo di elementi memorizzabili dipende dall'implementazione scelta e l'ammontare delle risorse ancora libere sull'FPGA. Le dimensioni della coda possono assumere alcuni valori fissi, perché il compilatore può implementare efficacemente sul target solo determinate estensioni. Per l'implementazione si possono scegliere tre risorse differenti:

- Flip Flop presenti nelle Slice M dell'FPGA. Conviene utilizzare questa tecnica per avere migliori performance ma per FIFO molto piccole (dimensioni minori di 100bytes)
- LUT, utilizzando le sei presenti in ogni Slice. Anche in questo caso, conviene utilizzare questa soluzione nel caso di FIFO piccole (minore di 300bytes)
- Blocchi di memoria. I Flip Flop e le LUT vengono preservati per l'implementazione dei VI da eseguire su FPGA.

Anche in questo caso sussiste la differenza tra VI-Scoped e Target-Scoped. La prima modalità permette di utilizzare, una singola struttura FIFO, per il trasferimento dati tra cicli nello stesso VI. La seconda permette, ad una singola FIFO, il trasferimento di dati tra multipli VI sullo stesso target.

Per servirsi delle FIFO si utilizza la funzione FIFO Method, presente nella paletta Memory & FIFO. Premendo il pulsante destro nel mouse su FIFO Method, è possibile scegliere, dal menu a tendina Select FIFO come mostrato in Figura 2.45, la FIFO su cui operare. Scegliendo invece Select

Method, si può selezionare quale operazione deve essere eseguita, tra la lettura e la scrittura, sulla FIFO.

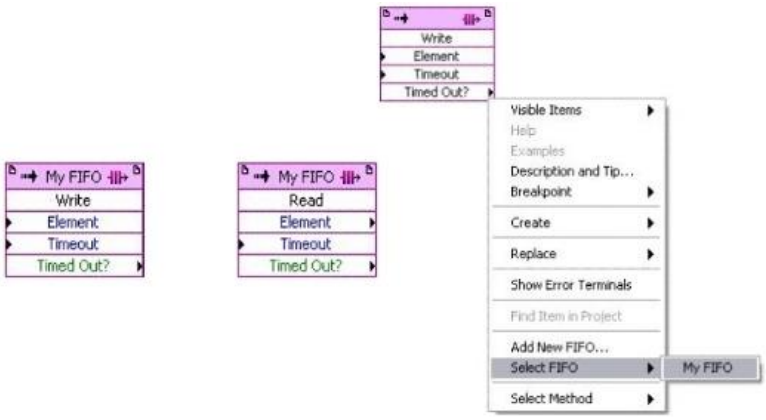

Figura 2.45 Funzioni di gestione delle FIFO

Nelle funzioni di lettura e scrittura sono presenti i terminali in cui indicare l'elemento da leggere o da scrivere. Inoltre, ci sono i terminali di Timeout, in cui assegnare il numero di tick che la funzione deve attendere per uno spazio libero se la FIFO è piena, e il Timed Out? in cui si indica se l'operazione di lettura è riuscita o fallita.

Si può verificare una condizione di overflow nella FIFO, nel caso in cui il ciclo di scrittura sia più veloce del ciclo di lettura. In questo caso, si ottiene un Timed Out nel metodo di scrittura con una perdita del dato da scrivere. Lo spazio, in una coda FIFO, può essere creato soltanto eseguendo la lettura o il reset sulla coda. I dati saranno persi, se occorre una condizione di Timed Out, fin quando non sarà reso disponibile uno spazio libero. La situazione opposta, in cui il ciclo di scrittura risulta più lento del ciclo di lettura, viene indicato con il termine underflow. In questo caso la coda FIFO rimarrà vuota e il metodo di lettura potrebbe dare un Timed Out.

Per evitare questi problemi si utilizzano altri due metodi disponibili: Get number of elements to Read e Get number of Elements to Write. Questi metodi consentono, rispettivamente, di conoscere il numero di elementi rimanenti nella FIFO da leggere e il numero di elementi rimanenti disponibili per la scrittura senza il verificarsi di un Timed Out.

Integrazione tra VI FPGA e VI Host

La comunicazione, tra un programma su FPGA e uno eseguito sul processore, viene utilizzata per il trasferimento e il salvataggio di dati, per l'interfaccia utente e per effettuare delle operazioni non implementabili in FPGA.

Per creare un programma eseguito sul processore, chiamato **VI Host**, si deve realizzare un VI sotto My computer, nella finestra di esplorazione del progetto. È possibile effettuare questa operazione, come è mostrato in Figura 2.46, premendo il pulsante destro del mouse su My Computer e selezionando New e VI.

Figura 2.46 Creazione di un programma da eseguire sul processore

Una volta creato il VI Host, si crea un'interazione tra i due VI (quello sull'host e quello sull'FPGA) inserendo delle funzioni presenti nella paletta FPGA Interface. Con queste funzioni, mostrate in Figura 2.47, è possibile:

- stabilire una connessione tra i programmi su FPGA e sul processore
- scrivere e leggere dati dall'FPGA
- controllare il flusso di esecuzione del VI sull'FPGA
- terminare la comunicazione tra i due programmi.

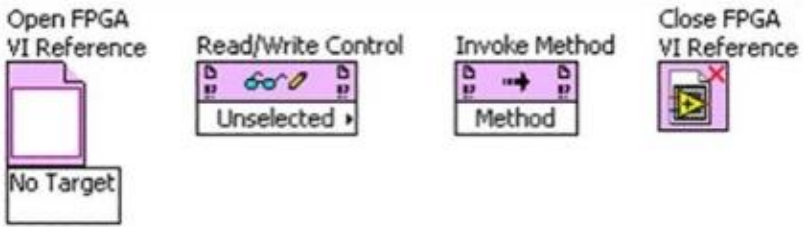

Figura 2.47 Funzioni per la comunicazione tra l'FPGA e il processore

Per stabilire una connessione tra i due programmi si utilizza la funzione **Open FPGA VI Reference**. Aprendo il box di configurazione di questa funzione, è possibile scegliere il programma da utilizzare tra un VI da compilare o un Bitstream già compilato. È sempre necessario aprire il riferimento per creare un canale di comunicazione tra il VI Host e il VI FPGA.

Le funzioni **Read/Write Control** permettono, una volta aperto il riferimento, di leggere i valori dagli indicatori o di scrivere dei valore sui controlli, presenti nel Front Panel del VI FPGA.

Per invocare un metodo o un'azione, dal VI Host, si utilizza **Invoke Method**. I metodi utilizzabili con queste funzioni sono, per esempio, Run, Abort, Reset, Download e le operazioni di lettura e scrittura sulle FIFO DMA.

L'ultima funzione in Figura 2.47 è la **Close FPGA VI Reference.** Con questa funzione si chiude il riferimento, si ferma l'esecuzione dell'FPGA e se ne effettua il reset riportando i valori alle condizioni iniziali. La funzione offre all'utente la libertà di scegliere se semplicemente chiudere il riferimento o di effettuare anche il reset dei valori.

I due VI, su host e su FPGA, sono completamente asincroni, ognuno viene eseguito con una propria velocità e indipendentemente. Il trasferimento dei dati può essere implementato in due modi differenti. Vengono usati i controlli sul Front Panel se l'applicazione ha bisogno del passaggio di singoli valori, e non di un insieme di dati. Se si ha bisogno del trasferimento di un'alta quantità di dati, si usano delle strutture di buffering dei dati.

Il miglior modo per usufruire del buffer è di utilizzare un DMA, un dispositivo hardware dedicato al trasferimento dei dati. L'utilizzo di questo dispositivo permette di scambiare una quantità rilevante di dati, tra l'FPGA e la memoria del computer, senza l'uso del processore. Il DMA ha bisogno solo di tre indirizzi per trasferire anche una quantità rilevante di dati. Quindi, per il trasferimento, vengono utilizzati una FIFO sull'FPGA, un DMA e delle locazioni di memoria RAM del controller.

Per esempio, in una comunicazione dal target FPGA al sistema Host, un primo trasferimento avviene, elemento per elemento, in una coda FIFO implementata direttamente in FPGA. Questo permette un primo impacchettamento dei dati in una coda di ridotte dimensioni (la dimensione massima di elementi per la coda realizzata nel dispositivo PXIe-7965, utilizzando dei dati unsigned a 16bit, è di 32767). In seguito un DMA esegue il trasferimento dei dati verso la memoria RAM del processore, senza il suo intervento. I dati vengono posti all'interno di un'allocazione di memoria, organizzata come una coda FIFO, presente nella RAM del controller. I dati presenti sulla RAM possono essere letti dal VI eseguito sul processore che può elaborare i dati o memorizzarli. Nella Figura 2.48 viene mostrato graficamente il trasferimento dei dati.

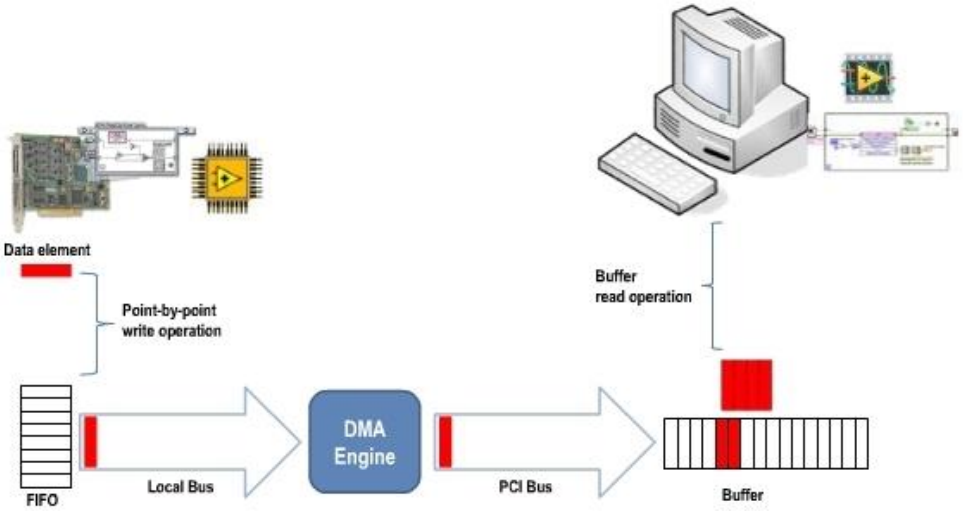

Figura 2.48 Realizzazione di una coda FIFO per il trasferimento dati
dal target al processore su FPGA e sulla memoria RAM

È importante, ai fini di un'implementazione efficace, capire le velocità di trasferimento del DMA e la velocità di accesso ai dati del VI Host.

All'utente viene data la possibilità di definire la dimensione della coda da implementare su FPGA, utilizzando i blocchi di RAM, già predisposti, per la realizzazione di una FIFO. Le dimensioni della coda possono assumere alcuni valori perché il compilatore può implementare solo determinate estensioni.

Un altro parametro configurabile è la dimensione della FIFO implementata nella RAM del controller. L'utente può scegliere la dimensione migliore, all'interno della funzione di lettura della coda, per non perdere i dati. Se si sceglie una dimensione piccola, si espone il sistema al problema di poter perdere molti campioni perché il VI Host non sarà in grado di leggerli tutti. Nel caso in cui si sovradimensiona la coda sulla RAM, si rischia di dover elaborare una quantità elevata di dati e di perdere anche in questo caso degli elementi.

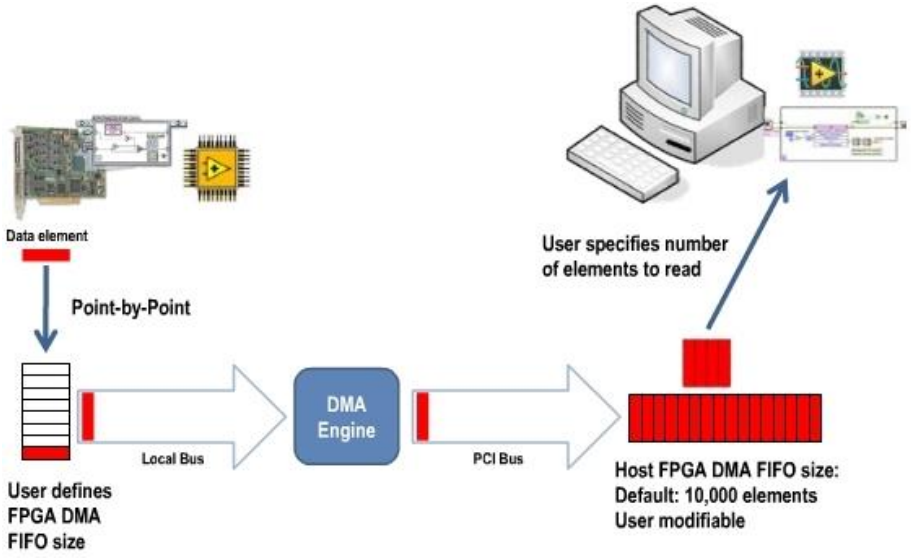

Figura 2.49 Parametri personalizzabili nel trasferimento tra FPGA e Host

Se accade una condizione di overflow, per una velocità di scrittura maggiore rispetto a quella di lettura, si possono apportare dei miglioramenti per eliminare il problema. Si può ridurre la velocità di scrittura dei dati nella FIFO, incrementare il numero di elementi letti, incrementare la grandezza del buffer sia su FPGA che su Host o ridurre il carico della CPU.

Si potrebbe verificare anche il caso di underflow, in cui la FIFO è svuotata a causa di una velocità di lettura più alta rispetto a quella di scrittura, generando un Timed Out nella lettura della coda. Per risolvere il problema si può andare ad incrementare il tempo di Timed Out, legge meno frequentemente o leggere dei pacchetti di dati più piccoli.

La creazione di una **FIFO DMA** è realizzata in maniera identica alla creazione di una FIFO per il trasporto dei dati all'interno del VI FPGA. Si può creare una nuova FIFO, premendo il pulsante destro sul target e selezionando new e poi FIFO. Questo permette di aprire la finestra di configurazione della FIFO in cui si può scegliere la rappresentazione degli elementi, la lunghezza della coda e la sua modalità. La scelta della modalità differisce nella creazione delle due FIFO. Nel caso di condivisione dei dati all'interno del VI FPGA, si sceglieva Target-Scoped, invece nel caso di trasferimento tra l'FPGA e il VI Host si deve scegliere Target to Host – DMA oppure Host to Target – DMA, in base alla direzione da dare al collegamento.

Per la scrittura e la lettura sulla FIFO nel VI Host si devono usare i blocchi Invoke Method. Per ogni blocco Invoke Method si deve selezionare sia la coda FIFO DMA e, in base alla direzione del collegamento, la funzione Write o Read (se la modalità della FIFO è Target to Host vi sarà la

funzione Read, se Host to Target ci sarà Write). Nel VI FPGA si utilizzano i blocchi per la lettura e la scrittura della FIFO, presenti nella palette Memory & FIFO.

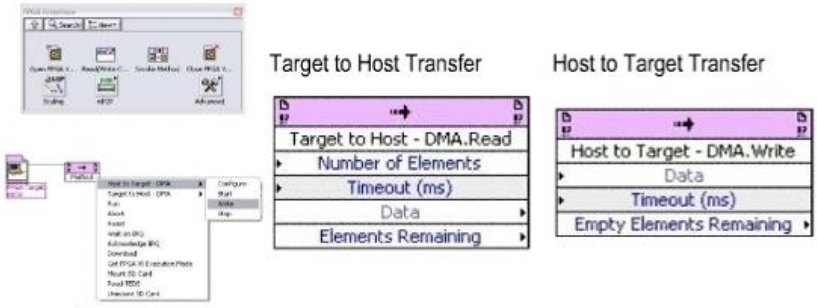

Figura 2.50 Funzioni per la gestione delle code FIFO nel VI Host,
la scelta del metodo e la loro interfaccia

3. Implementazione Del sistema di analisi

Organizzazione e distribuzione delle operazioni tra FPGA e Host

In questo capitolo vengono presentate le fasi di progetto e di realizzazione, in Labview 2012, dell'Echo Generator. L'EG deve implementare quattro operazioni fondamentali:

- Consentire all'utente di controllare e monitorare le funzioni svolte dal sistema.
- L'acquisizione e la memorizzazione, su disco, del segnale generato dal radar. Il segnale sarà prima traslato in banda base e demodulato nelle componenti I&Q, dall'RF Front End e dall'Up-Down Conversion, per consentire l'acquisizione digitale dell'impulso del radar.
- Il calcolo delle componenti I&Q del segnale riflesso dal suolo marziano, come risposta all'impulso inviato dal radar. Il segnale riflesso varierà in ampiezza, forma e ritardo temporale dall'invio dell'impulso, in base alla posizione del radar.
- La generazione delle componenti I&Q con l'Arbitrary Waveform Generator (AWG).

La prima operazione, nella fase di progettazione, consiste nella distribuzione delle operazioni da eseguite tra i vari sistemi hardware presenti. Il processore permette di implementare le operazioni a bassa priorità, come la memorizzazione dei dati su disco, o l'interfaccia utente per il controllo del programma. Sull'FPGA, invece, verranno eseguite le operazioni ad alta priorità, e le interfacce con i vari segnali analogici, grazie agli adapter module connessi alle FPGA. Quindi, sul processore sarà eseguito un programma, chiamato VI Host, che fornirà l'interfaccia con l'utente ed eseguirà le operazioni di controllo e monitoraggio dei VI FPGA. Attraverso il trasferimento dei dati dall'FPGA al processore con delle FIFO DMA, il VI Host permetterà la memorizzazione dei dati su disco. L'FPGA eseguirà l'acquisizione dell'impulso inviato dal radar, il calcolo dell'impulso riflesso e l'invio dei dati all'AWG per la generazione. L'AWG viene configurato dal programma Host, ma in seguito viene controllato direttamente dall'FPGA per il trasferimento dei dati e l'invio del trigger per la generazione. Sull'FPGA viene eseguita una prima formattazione dei dati da trasferire su disco, per alleggerire il carico e le operazioni da eseguire sull'Host.

La versione finale, del progetto realizzato, viene mostrato in Figura 3.1.

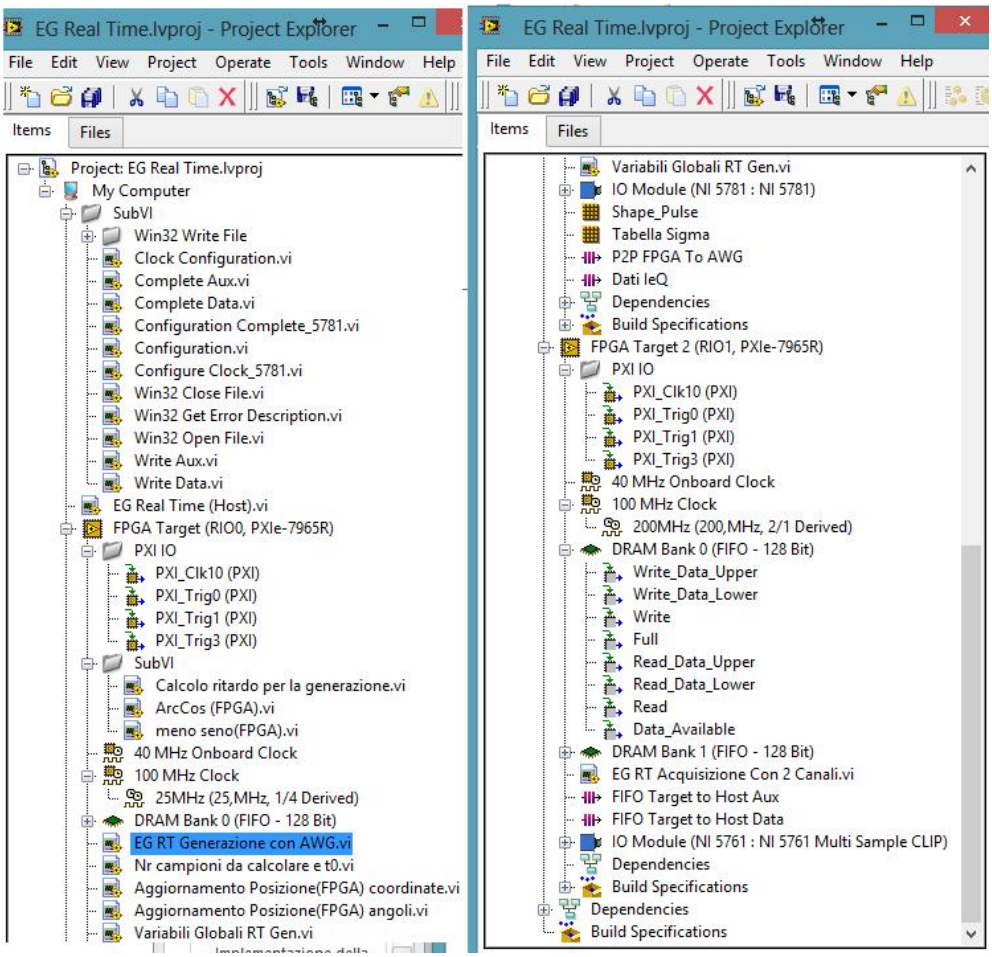

Figura 3.1 Versione finale del progetto EG Real Time

Nella finestra di esplorazione del progetto, sono raccolti tutti i programmi per la realizzazione dell'EG. I VI realizzati sono:

- "EG Real Time(Host)", eseguito sul processore, consente di richiamare gli altri VI creati e di realizzare la connessione con il VI eseguiti su FPGA. Questo programma fornisce l'interfaccia utente del sistema
- "EG RT Acquisizione Con 2 Canali", eseguito su FPGA, effettuata la temporizzazione del sistema, l'acquisizione dai segnali analogici con il modulo adapter NI 5761 utilizzato
- "EG RT Generazione con AWG", eseguito su FPGA, effettuata la temporizzazione del sistema, il calcolo delle componenti I&Q, il controllo e il trasferimento dei dati all'AWG per la generazione
- I SubVI "Configuration", "Clock Configuration", "Configuratin Complete_5781" e "Configure Clock_5781" che permettono una corretta inizializzazione dei moduli adapter.
- I SubVI "Arccos(FPGA)", "Calcolo Ritardo per la Generazione", "Nr Campioni da calcolare e T0" eseguiti su FPGA, che permettono una riduzione del codice visualizzato nei VI FPGA principali.
- I SubVI "Write Aux", "Write Data" , "Complete Data" e "Complete Aux" utilizzati per la scrittura dei dati su disco

Nella scelta dei componenti hardware, è stato necessario l'utilizzo di due FPGA. L'utilizzo di due unità FlexRio permette di migliorare la velocità di trasferimento dei dati. Questa velocità viene limitata, in teoria, a 800MB/s per direzione dal Bus PCIe del PXI-1082.
Il limite diminuisce, di circa -10% o -15%, se la scheda FlexRIO esegue un trasferimento di dati sia in ingresso che in uscita. Quindi nel caso in cui sia utilizzata un'unica FPGA, come mostrato nella Figura 3.2 a sinistra, il limite della velocità per il trasferimento dei dati dall'FPGA all'AWG, permesso dal sistema, sarà di 760MB/s. I restanti 40MB/s verranno utilizzati per il trasferimento dei dati acquisiti su FPGA verso il processore. Nel nostro caso, è richiesta una velocità di 800MB/s solo per il trasferimento dei dati dall'FPGA all'AWG, dovuto ad una frequenza di campionamento di 200MS/s da 16 bit e l'utilizzo dei due canali per la generazione delle componenti I&Q.

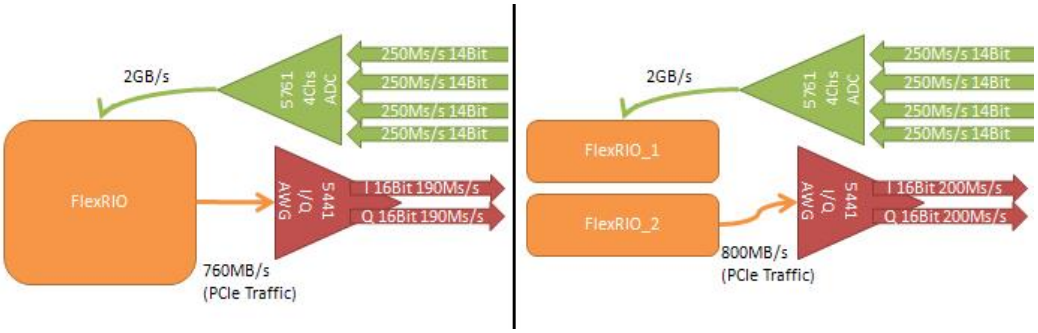

Figura 3.2 Confronto tra due possibili configurazioni del sistema

L'adapter module NI 5761, collegato direttamente all'FPGA, permette un trasferimento continuo di dati a 2GB/s verso l'FPGA. Nel caso in cui siano utilizzate due FPGA, come mostrato nella Figura 3.2 a destra, il limite della velocità per il trasferimento dei dati dall'FPGA all'AWG passa a 800MB/s. In questo caso, il requisito per la velocità di trasferimento viene soddisfatto.

Nell'FPGA utilizzata per il calcolo dei campioni da generare, saranno implementati due algoritmi. Uno per l'aggiornamento dei dati cinematici e il calcolo delle componenti I&Q, l'altro per il calcolo dell'eco riflesso normalizzato. Nei paragrafi seguenti, vengono presentate le tecniche per implementare gli algoritmi e per migliorare la velocità e la precisione dei calcoli.

Gerarchia dei programmi

La gerarchia in Figura 3.3, indica quali SubVI vengono richiamati all'interno del VI Host.

La figura mostra, inoltre, le librerie utilizzate dal VI Host per interfacciarsi con i VI FPGA, le funzioni per la gestione degli errori e le librerie di Windows per le operazioni di scrittura su disco. Inoltre, mostra i programmi utilizzati per la gestione dell'AWG e delle code Peer To Peer, impiegate per il trasferimento dei dati.

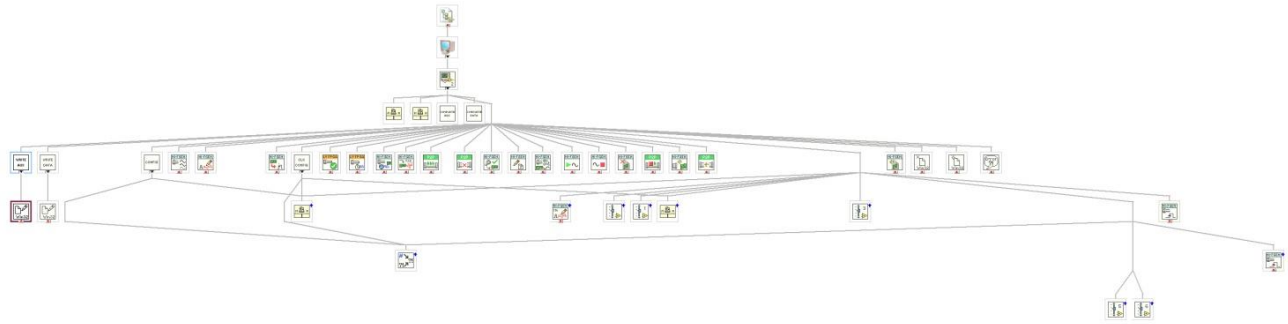

Figura 3.3 Gerarchia dei programmi utilizzati

Il VI Host, che si può notare nel secondo livello partendo dall'alto, richiama gli altri SubVI per la scrittura su disco, per la comunicazione con le FPGA e per la configurazione dell'AWG.

Algoritmo per l'aggiornamento dei dati cinematici e il calcolo delle componenti I&Q

La prima parte delle elaborazioni prevede l'aggiornamento dei dati cinematici e il calcolo della potenza reale dei campioni. Queste operazioni devono essere eseguite ad ogni impulso di PRI, differentemente dell'aggiornamento dell'eco riflesso normalizzato. Dovendo eseguire più fasi elaborative, l'esecuzione di questa porzione di codice è stata divisa in più parti per semplificare l'analisi e l'implementazione. Nella prima parte del codice, mostrato in Figura 3.5, viene eseguito l'aggiornamento dei dati cinematici. Sono utilizzate delle variabili d'appoggio per aggiornare i valori, senza perdere i dati ricevuti dal simulatore dell'interfaccia dell'EDM. Sono state impiegate delle variabili globali, invece di semplici controlli, perché nel linguaggio Labview FPGA rappresentano la virtualizzazione di registri fisici. Se si utilizzano dei semplici controlli, nella compilazione questa scelta viene interpretata come quella di voler rendere accessibili i valori assunti dal controllo al VI Host. Quindi crea un collegamento con il VI Host, non necessario, rallentando l'esecuzione del VI FPGA. L'aggiornamento dei dati cinematici viene eseguito presupponendo un moto uniforme del radar. Le l'espressione utilizzata è rappresentata nell'Equazione 3.1.

$$Z_n = Z_{n-1} + V_z * t_{PRI}$$

Equazione 3.1

Dove:

- Z_n rappresenta il valore aggiornato della variabile
- Z_{n-1} è il valore, della variabile, assunto al ciclo precedente
- V_z è il valore della velocità del Radar
- t_{PRI} rappresenta il tempo trascorso tra due impulsi di PRI

Nella Figura 3.5, sono presenti sei sottoprogrammi per permettere l'aggiornamento delle variabili X,Y, Z, α, β e γ. All'interno dei sottoprogrammi, tutti uguali tra loro, viene eseguito il codice mostrato in Figura 3.4.

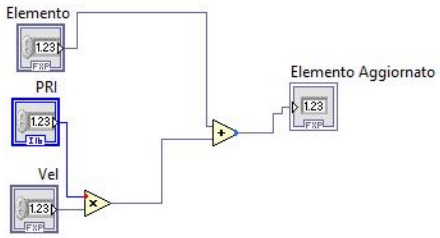

Figura 3.4 SubVI utilizzato per l'aggiornamento di un dato cinematico

Quindi, in Figura 3.5, viene mostrato l'intero codice, per eseguire l'aggiornamento, oltre alle funzioni Tick Count, che permettono di valutare il periodo di esecuzione.

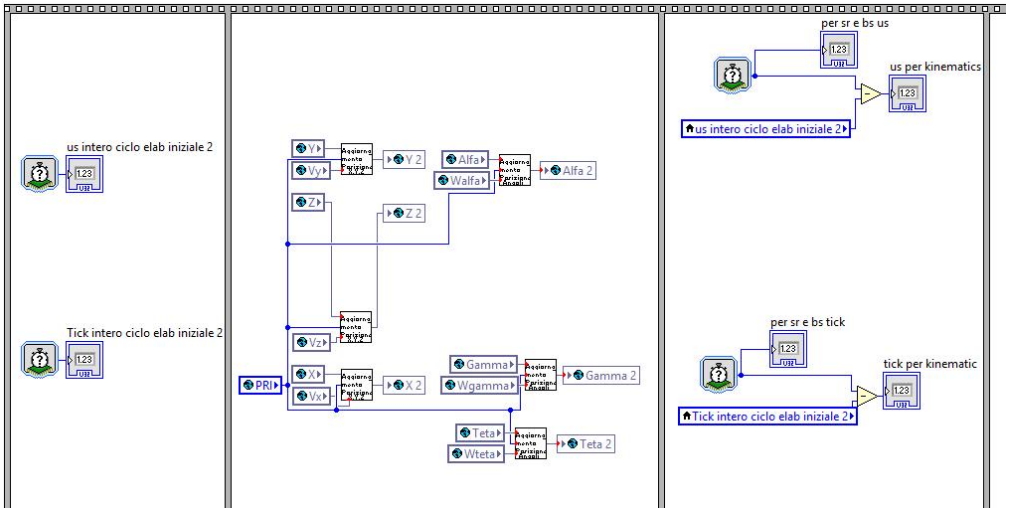

Figura 3.5 Codice utilizzato per l'aggiornamento dei dati cinematici e funzioni per valutare il costo computazionale

Nella scrittura del codice si deve tener conto, inoltre, della possibilità di dover aggiornare i valori nelle variabili temporanee con nuovi valori ricevuti dal simulatore dell'interfaccia con il modulo EDM. Il codice mostrato nella Figura 3.6, permette di aggiornare i valori solo quando vengono ricevuti nuovi dati cinematici dal simulatore. Questa porzione di codice viene eseguita in contemporanea con l'avvio del calcolo di un nuovo eco normalizzato.

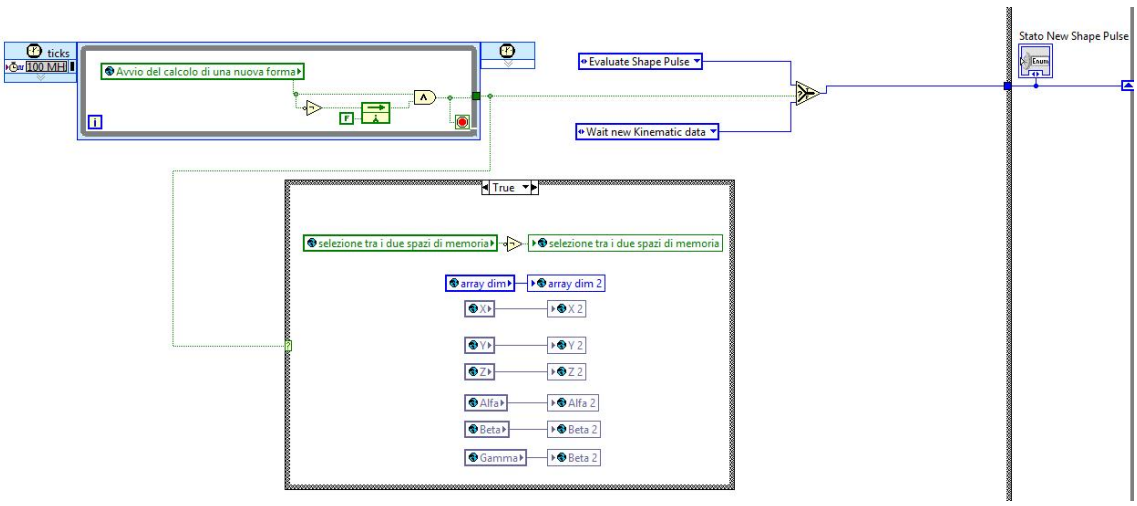

Figura 3.6 Codice per utilizzare i dati cinematici aggiornati, ricevuti dal simulatore dell'EDM

La parte successiva del codice, in Figura 3.7, permette di calcolare l'angolo di off-nadir e la quota, chiamata Slant Range, del radar. L'espressione per il calcolo dell'angolo di off nadir, mostrata nell'Equazione 3.2, è stata estratta dai documenti che riportano le specifiche dell'EG.

$$\theta_{off-nadir} = \cos^{-1}(-\sin\beta)$$

Equazione 3.2

Lo Slant Range viene calcolato come

$$SR = \frac{Z}{-\sin\beta}$$

Equazione 3.3

Nella Figura 3.7, viene riportato il codice per il calcolo dello Slant Range e dell'angolo di off-nadir, oltre alle funzioni Tick Count, utilizzate per valutare il periodo di esecuzione.

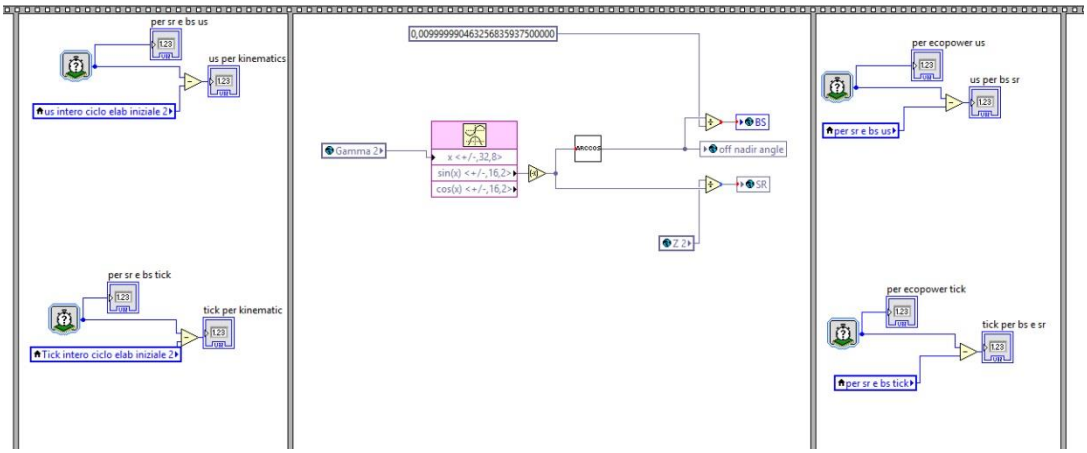

Figura 3.7 Codice per il calcolo dello Slat Range, dell'angolo di off-nadir, oltre alle funzioni per valutare il costo computazionale

La funzione $\cos^{-1}(x)$, comunemente definita arccos(x), è stata implementata attraverso un SubVI. Il valore risultante viene calcolato utilizzando uno sviluppo in serie di Taylor al terzo ordine, mostrato nell'Equazione 3.4. Nella Figura 3.8 viene mostrato il codice Labview per descrivere l'Equazione 3.4

$$\cos^{-1} x = \frac{\pi}{2} - x - \frac{x^3}{6}$$

Equazione 3.4

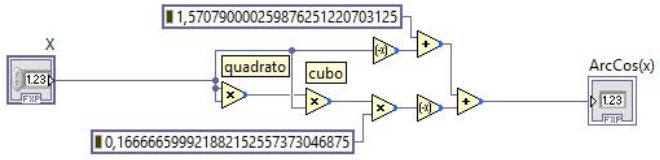

Figura 3.8 Codice dello sviluppo in serie, al terzo ordine, dell'arccos(x)

La fase successiva consiste nella valutazione della potenza reale del segnale. Questo parametro viene usato nell'elaborazione successiva dei campioni. L'espressione utilizzata per il calcolo della $P_{r,real}$ è mostrata nell'Equazione 3.5.

$$P_{r,real} = \frac{10^{-7} P_{TX} \sigma_0 H^2}{SR^4}$$

Equazione 3.5

Nel nostro caso, P_{TX} risulta pari ad 1W e il valore di σ_0, il coefficiente di backscattering, viene estratto da una tabella, in cui sono contenuti i valori assegnati al coefficiente. L'indice per la selezione dell'elemento sezionato dalla tabella viene calcolato come mostrato nell'Equazione 3.6.

$$indice_\sigma = round\left[\frac{\theta_{off-nadir}}{\delta_\theta}\right]$$

Equazione 3.6

Per il calcolo della potenza reale sono state utilizzate le funzioni aritmetiche pipelined. Queste funzioni saranno presentate, con maggiore dettaglio, nel paragrafo Pipeline. Non è necessario l'uso di elementi di ritardo nell'elaborazione, perché solo per l'operazione di divisione presenta una latenza iniziale. In Figura 3.9 viene mostrato il codice per implementare il calcolo.

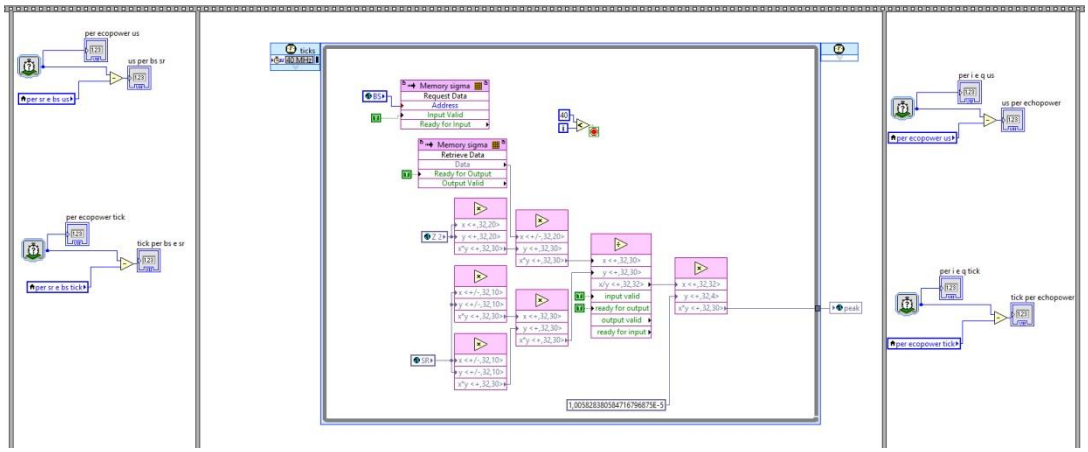

Figura 3.9 Codice per il calcolo di $P_{r,real}$ e per valutare il costo computazionale

Vengono eseguiti 40 cicli, necessari per caricare la serie dei registri della divisione pipelined (33 registri che corrispondono a 33 cicli del timed loop), e per leggere correttamente il parametro σ_0 dalla tabella.

Successivamente vengono calcolate le componenti I&Q del segnale da generare. Le espressioni, utilizzate per il calcolo delle componenti, vengono mostrate nelle Equazione 3.7 e Equazione 3.8.

$$I = \sqrt{100 P_r}\cos(1{,}49895 \cdot 10^3 \cdot SR)$$

Equazione 3.7

$$Q = \sqrt{100 P_r}\sin(1{,}49895 \cdot 10^3 \cdot SR)$$

Equazione 3.8

Dove

$$P_r = P_{r,real} \cdot P_{r,norm}$$

Equazione 3.9

In questo caso $P_{r,real}$ rappresenta la potenza reale del segnale, calcolata nel frame precedente, e $P_{r,norm}$ il campione dell'eco normalizzato. Il campione di $P_{r,norm}$ verrà letto dalla memoria Shape_Pulse, la quale conterrà i campioni dell'eco calcolati in un altro ciclo. Attraverso la variabile "selezione tra i due spazio di memoria", è possibile selezionare l'indirizzo da cui leggere i campioni. Vengono utilizzati due spazi di memoria per permette al VI FPGA di calcolare un nuovo eco normalizzato e utilizzare quello calcolato nell'iterazione precedente. Si può notare, dalla Figura 3.10, che il calcolo viene eseguito in un Single-Cycle Timed Loop e viene utilizzato un clock a 25MHz. La frequenza di esecuzione viene limitata, non dal codice presente in questo ciclo, ma da quello presente nel ciclo per il calcolo dell'eco normalizzato. Questo perché l'accesso alla memoria Shape_Pulse è consentita un unico dominio di clock, per evitare la corruzione dei dati. Il valore di 25MHz viene imposto dal calcolo dell'eco normalizzato. Il calcolo delle componenti I&Q viene eseguito in parallelo per due campioni, in modo da migliorare la velocità di esecuzione. Questo permette di sfruttare completamente il parallelismo della coda FIFO target-scoped, utilizzata per il trasferimento dei campioni. In ogni ciclo di esecuzione vengono calcolati quattro elementi (due coppie I&Q) da 16 bit, che uniti, permettono di utilizzare una coda con elementi formati da 64bit.

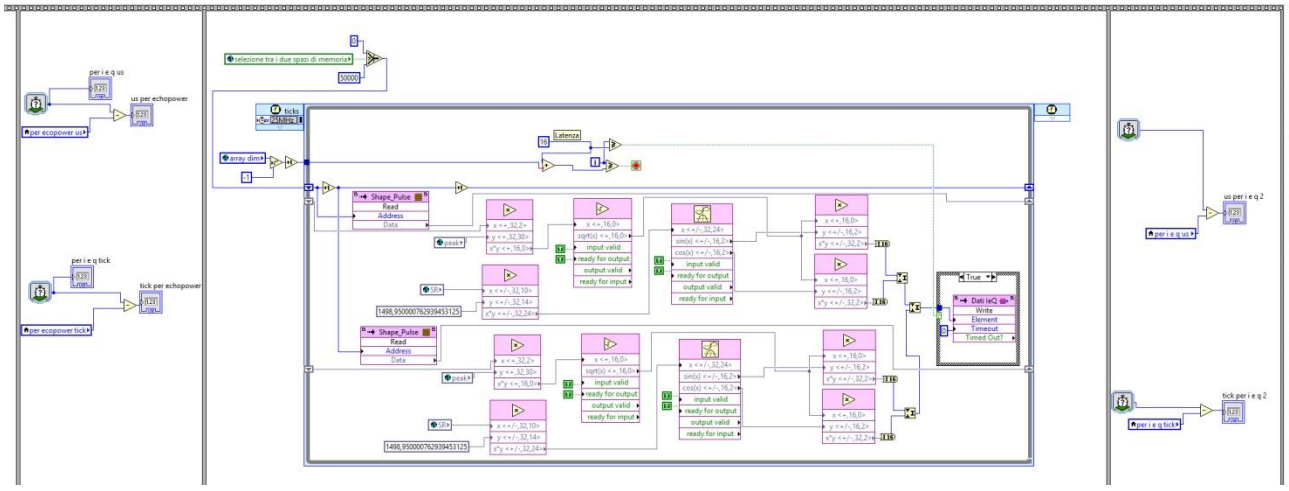

Figura 3.10 Codice per il calcolo delle componenti I&Q e per valutare il costo computazionale

Questo approccio permette di eseguire l'algoritmo nella metà dei cicli di clock, rispetto ad un'elaborazione per singolo campione. Utilizzando le funzioni pipelined, è presente una latenza iniziale di 16 cicli per ottenere in uscita dei campioni validi. La latenza è dovuta all'utilizzo della funzione per il calcolo del coseno e del seno.

Costo computazionale

Il costo computazionale del codice può essere valutato con le funzioni tick count. Nelle immagini precedenti, sono state mostrate le elaborazioni, unite alle funzioni per valutare il tempo di esecuzione. Le porzioni di codice mostrate nel paragrafo precedente verranno eseguiti in maniera sequenziale e ad ogni PRI. Dalle specifiche dell'EG, si possono trovare tre valori della PRI

corrispondenti a tre posizioni limiti del radar. La posizione limite viene definita con i valori dello Slant Range, ossia della quota del radar. Il valore dello Slant Range dipendono dalla distanza misurata sull'antenna altimetro, indicata come coordinata Z, e dall'angolo β o dall'angolo di off-nadir, aggiornati nel precedente PRI. Quindi, si possono trovare due casi distinti per uno stesso valore di Slant Range, una per la coordinata Z massima, e l'altra per l'angolo di off-nadir massimo. Nella Figura 3.11 vengono mostrati i tre casi possibili.

Valore limite di Sland Range	391m		6241m		8500m	
Distanza tra due PRI	10μs		46 μs		88 μs	
Valori limiti della coordinata Z	391m	224m	6241m	3580m	8500m	4875m
Valore limite dell'angolo di off nadir	0,05rad	0,8rad	0,05rad	0,8rad	0,05rad	0,8rad

Figura 3.11 Casi particolati di Slant Range e relativi periodi di PRI

Il caso peggiore, per ogni PRI, corrisponde al valore massimo di Sland Range. In questo caso, la dimensione dell'eco riflesso diventa la massima possibile per il corrispondente PRI. È stata eseguita una valutazione pratica dei tempi di esecuzione, mostrata in Figura 3.12, per ogni parte del codice presentato nel paragrafo precedente. Il costo computazionale per l'aggiornamento dei dati cinematici, per il calcolo dello Slant Range e del coefficiente di backscattering, per il calcolo della potenza reale non dipendono dalla dimensione dell'eco normalizzato. Quindi, queste parti non dipendono dallo Slant Range ma rimangono fisse per qualsiasi posizione del radar.

Porzione di codice considerata	Costo computazionale [μs]	Costo Computazionale [cicli di clock a 40MHz]
l'aggiornamento dei dati cinematici	0,25	10
Slant Range e del coefficiente di backscattering	2,5	100
potenza reale dell'eco riflesso	1,175	47

Figura 3.12 Costo computazionale per l'aggiornamento dei dati cinematici, per il calcolo dello Slant Range, del coefficiente di backscattering e per il calcolo della potenza reale

La durata dell'esecuzione del codice per le componenti I&Q, invece, cambia in base alla dimensione dell'eco, perché le operazioni vengono eseguite su ogni campione. In Figura 3.13,viene mostrata la dimensione dell'eco e il costo computazionale per il calcolo di I&Q rilevato.

Coordinata Z Angolo di off nadir	391m 0,05rad	224m 0,8rad	6241m 0,05rad	3580m 0,8rad	8500m 0,05rad	4875m 0,8rad
Dimensione eco normalizzato	102	84	1628	1338	2218	1822
Calcolo I&Q	2,725μs	2,375μs	33,25μs	27,45μs	45,05μs	37,125μs

Figura 3.13 Costo computazionale per il calcolo delle componenti I&Q

È possibile utilizzare il periodo precedente di PRI, per eseguire l'intero calcolo, perché il primo eco, da generare e acquisire, è solo rumore. Se il contributo, sommato, delle quattro parti risulta minore del valore del periodo di PRI, sarà possibile utilizzare praticamente il codice. Nella Figura 3.14 si può osservare il rispetto delle specifiche.

Coordinata Z	391m	224m	6241m	3580m	8500m	4875m
Angolo di off nadir	0,05rad	0,8rad	0,05rad	0,8rad	0,05rad	0,8rad
Periodo di PRI	10	10	46	46	88	88
Costo computazionale totale	6,65	6,3	37,175	31,375	48,975	41,05

Figura 3.14 Costo computazionale totale dell'algoritmo

In tutti i casi, l'elaborazione non supera il periodo limite del PRI. Quindi questa versione del codice può essere utilizzata per la descrizione del calcolo su FPGA.

Logica utilizzata

La logica utilizzata per implementare il programma può essere estratta dallo XilinxLog. Lo XilinxLog è un file in cui vengono descritte le operazioni eseguite dal compilatore Xilinx. Una descrizione migliore dello XilinxLog e delle opzioni di ottimizzazione dei VI FPGA, viene presentata nell'ultimo paragrafo di questo capitolo. I dati estratti dallo XilinxLog, provengono dalla compilazione di un VI FPGA, in cui erano presenti le quattro elaborazioni. Le risorse utilizzate per implementare il codice sono:

- Slice Totali: il 44,2% (6505 di 14720)
- Slice con Registri (SLICEM): il 25,1% (14803 di 58880)
    - Usati come Flip Flop: 14803
- Slice con LUT (SLICEL): il 24,2% (14224 di 58880)
    - Usati come Logica: 13884 di 58880
    - Usati come memoria: 256 di 24320
        - Come RAM Dual Port: 132
        - Come Shift Register: 124
    - Usati come Route-thru esclusivo: 124
- DSP48: il 9,2% (59 di 640)
- Blocchi di RAM: il 44,3% (108 di 244)

Per capire quali strutture base sono state utilizzate, per implementare il codice, può essere estratto il report avanzato della sintesi HDL, dallo XilinxLog. In questo caso, sono stati utilizzati:

- RAM : 8
- Moltiplicatori : 13
- Addozionatori/Ssottrattori : 253
- Contatori : 35

- Accumulatori : 2
- Registri come Flip Flop : 14504
- Comparatori : 561
- Multiplexer : 2155
- Decoder : 2
- Macchine a Stati finiti : 21
- Xor : 181
- Register per le FSM : 9872
- Shift Register : 92

Un altro report importate è quello riguardante il timing. In questo report, viene evidenziata la frequenza massima cui può lavorare il codice. Per ogni parte di codice, eseguita con un clock unico, viene specificata la frequenza massima cui può lavorare. Il report:

- Clock Onboard 40 MHz: 40,00 MHz (42,33 MHz massimo)
- Clock 100 MHz: 100,00 MHz (105,91 MHz massimo)
- Clock 25MHz: 25,00 MHz (54,62 MHz massimo)
- TS_BusClk: 128,85 MHz (massimo)
- TS_SlowBusClk: 181,49 MHz (massimo)
- TS_DramClk200: 210,13 MHz (massimo)
- TS_DramClk200s90: 214,64 MHz (massimo)
- TS_IoRxClock: 333,33 MHz (massimo)
- TS_ClockGenXilinxV5x_TxDcm_TxHighSpeedClkDcm: 279,72 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxHighSpeedClkDcm: 450,25 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxLowSpeedClkDcm: 176,12 MHz (massimo)

Se, nel codice, non viene specificato il clock da utilizzare, sarà impiegato uno di default, scelto nella fase iniziale di configurazione dell'FPGA. Il valore di default iniziale è di 40MHz. Nel nostro caso, l'aggiornamento dei dati cinematici, il calcolo dello Slant Range, del coefficiente di backscattering e il calcolo della potenza reale sono eseguiti a 40MHz. Il valore è molto vicino al valore massimo, di 42,33MHz, estratto dal report. La parte di codice eseguita a 25MHz, per il calcolo delle sole componenti I&Q, potrebbe essere eseguita con una frequenza più alta. In questo caso, la frequenza sarà limitata dalla velocità d'accesso alla memoria Shape_Pulse, fissata dal codice per il calcolo dell'eco normalizzato.

Calcolo dell'Eco Riflesso Normalizzato

Le operazioni per il calcolo dell'eco riflesso sono mostrate nelle Equazione 3.10, Equazione 3.11, Equazione 3.12, Equazione 3.13 ed Equazione 3.14.

$$\mathrm{IR}^{\mathrm{Asym}}(\tau) = \mathrm{G}(\varepsilon) \qquad \tau \geq 0$$

Equazione 3.10

$$G(\varepsilon) = \exp\left[-1004{,}03\left(\frac{(\sin\xi - \varepsilon\cos\xi)^2}{1+\varepsilon^2}\right)\right]\cdot\sqrt{\frac{2\pi}{a+2b}}$$

Equazione 3.11

$$a = 1004{,}03\cdot\frac{\varepsilon\sin 2\xi}{1+\varepsilon^2}$$

Equazione 3.12

$$b = 1004{,}03\cdot\frac{\varepsilon^2\sin^2\xi}{1+\varepsilon^2}$$

Equazione 3.13

$$\varepsilon = \sqrt{\frac{c\cdot\tau}{Z}}$$

Equazione 3.14

Nella Figura 3.15, viene mostrata la prima versione del calcolo per valutare i campioni dell'eco riflesso normalizzato.

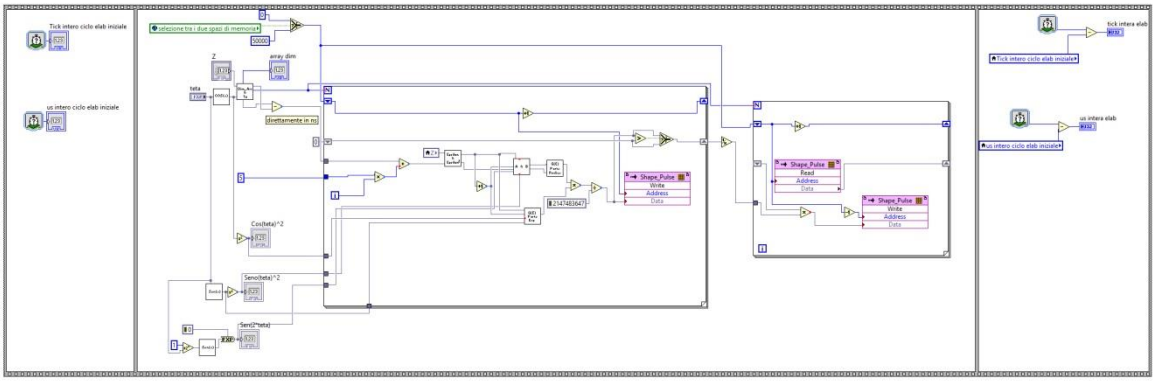

Figura 3.15 Prima versione del codice per il calcolo dell'eco normalizzato

Le prime operazioni eseguite sono il calcolo di:

- Cos(ξ) dove ξ rappresenta l'angolo di off-nadir, l'angolo formato tra la direzione in cui punta il radar e il nadir (la direzione ortogonale alla superfice osservata e passante per il radar)
- $Cos^2(\xi)$
- $Sen^2(\xi)$
- Sen(2ξ)
- Dimensione dell'array dell'eco riflesso, ossia il numero di campioni da calcolare per costruire l'intero eco riflesso
- Istante iniziale per il calcolo dell'eco riflesso, valutato come t_{ric}-t_0

Nella Figura 3.15, l'ingresso teta rappresenta l'angolo di off-nadir e z rappresenta la quota del radar.

I valori del seno e del coseno, in questa prima versione, vengono calcolati utilizzando un'approssimazione in serie di Taylor. Queste serie sono approssimate, per il coseno, al terzo e, per il seno, al quinto ordine.

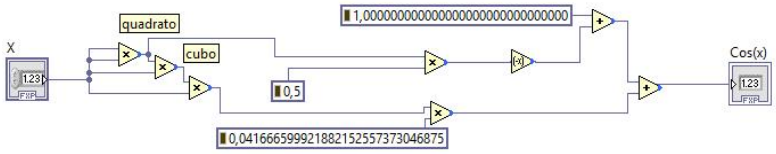

Figura 3.16 Implementazione dello sviluppo in serie di Taylor del coseno

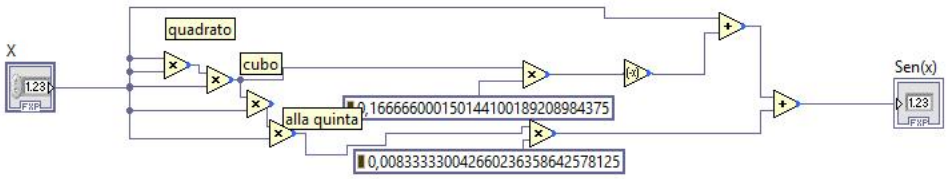

Figura 3.17 Implementazione dello sviluppo in serie di Taylor del seno

La dimensione e l'istante iniziale per il calcolo dell'eco riflesso vengono elaborati nel SubVI "Nr Campioni da calcolare e T0", mostrato nella Figura 3.18.

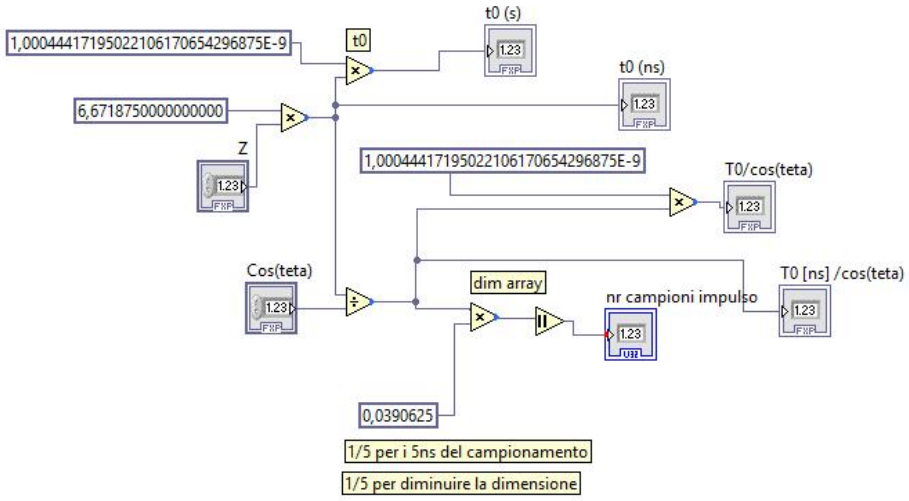

Figura 3.18 Codice per il calcolo della dimensione e dell'istante iniziale per l'eco riflesso

La dimensione può essere calcolata come il rapporto tra la differenza, tra il primo e l'ultimo istante in cui viene calcolato l'eco riflesso, e il periodo di generazione del segnale (5ns). Il vettore dei tempi è definito nell'Equazione 3.15.

$$\tau = \left[2\frac{Z}{c * \cos\xi} - 2\frac{Z}{c};\ 2\frac{Z}{c * \cos\xi} + t_{ck} - 2\frac{Z}{c}; \dots\dots\dots\dots;\ 4\frac{Z}{c * \cos\xi} - 2\frac{Z}{c}\right]$$

Equazione 3.15

La dimensione può essere calcolata come nell'Equazione 3.16.

$$\mathrm{Dim_{Array}} = \frac{4\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi} - 2\frac{\mathrm{Z}}{\mathrm{c}} - \left(2\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi} - 2\frac{\mathrm{Z}}{\mathrm{c}}\right)}{5 * 10^{-9}} = \frac{2\frac{\mathrm{Z}}{\mathrm{c} * \cos\xi}}{5 * 10^{-9}}$$

Equazione 3.16

Per facilitare l'implementazione, è stata sviluppata una versione semplificata dell'espressione, mostrata nell'Equazione 3.17. In questa versione viene aggiunto un fattore 1/5 per alleggerire il calcolo dell'eco normalizzato.

$$\mathrm{Dim_{Array}} = \frac{2\mathrm{Z}}{5 * 10^{-9} * 2.99792 * 10^{8} * \cos\xi} * \frac{1}{5} = \frac{6.67128 * Z}{25 * \cos\xi}$$

Equazione 3.17

Anche nel calcolo dell'istante iniziale, t_{ric}-t_0, viene eseguita una semplificazione. Il calcolo viene eseguito direttamente in ns.

$$(\mathrm{t}_{ric} - \mathrm{t}_0) * 10^9 = \frac{2 * \mathrm{Z} * 10^9}{2.99792 * 10^8 \cos\xi} - \frac{2 * \mathrm{Z} * 10^9}{2.99792 * 10^8} = \frac{6.67128 * \mathrm{Z}}{\cos\xi} - 6.67128 * \mathrm{Z}$$

Equazione 3.18

La dimensione dell'eco riflesso, del Cos(ξ), $Cos^2(\xi)$, $Sen^2(\xi)$, Sen(2ξ) e t_{ric}-t_0 sono calcolate solo una volta per ogni eco. Una volta valutati questi parametri, si può continuare con il calcolo dei campioni. Nella prima versione, è stato utilizzato un ciclo For. Viene calcolato, per ogni ciclo, un campione valutando i parametri ε, A, B nei vari istanti, utilizzando l'Equazione 3.12, l'Equazione 3.13 ed l'Equazione 3.14. Per il calcolo di ε si tiene in conto del fattore 10^9 utilizzato per il calcolo di t_{ric}-t_0. Quindi verrà utilizzata l'Equazione 3.19.

$$\varepsilon = \sqrt{\frac{0{,}299792 \cdot \tau}{Z}}$$

Equazione 3.19

Nella Figura 3.19 e Figura 3.20, vengono mostrati i codici utilizzati per implementare le valutazioni di ε, A e B.

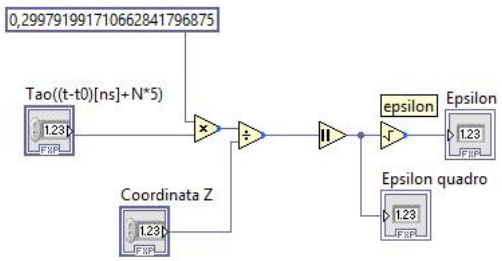

Figura 3.19 Codice Labview per il calcolo di ε

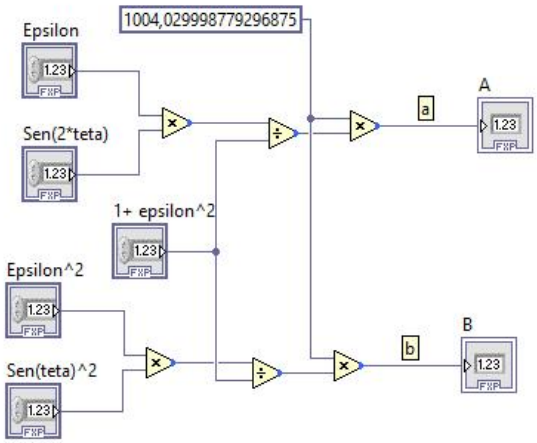

Figura 3.20 Codice Labview per il calcolo di A e B

Il calcolo della G(ε) viene diviso in due parti. La prima parte corrisponde al calcolo del primo fattore nell'Equazione 3.11, quello esponenziale. La seconda parte corrisponde al secondo fattore, quello sotto la radice quadrata. Per implementare la funzione esponenziale, è stato utilizzato un SubVI, in cui viene utilizzato lo sviluppo in serie di Taylor al terzo ordine.

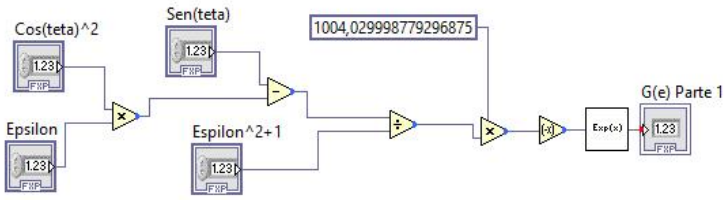

Figura 3.21Valutazione del fattore esponenziale della G(ε)

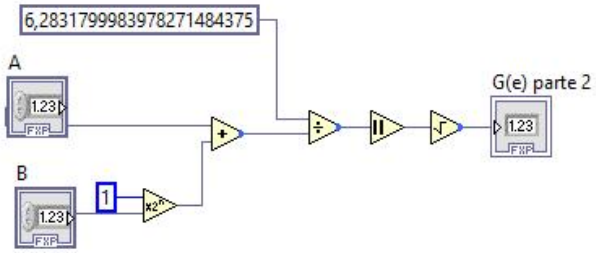

Figura 3.22 Valutazione del fattore sotto radice quadrata della G(ε)

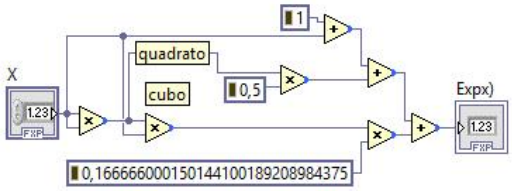

Figura 3.23 Implementazione dello sviluppo in serie di Taylor dell'esponenziale

Alcune funzioni, per esempio per divisione, ha bisogno di più cicli di clock per essere eseguita. Per questo, non è possibile utilizzare un Single-Cycle Timed Loop per forzare il codice ad essere

eseguito in un unico ciclo di clock. Utilizzando un For Loop sono necessari più cicli di clock per il calcolo di un campione dell'eco normalizzato.

Costo computazionale

Utilizzando questa implementazione, sono necessari un numero di cicli elevato per completare l'elaborazione. Nella Figura 3.24 vengono mostrati i tempi di esecuzione necessari per il calcolo dell'eco normalizzato. I tempi di esecuzione vengono espressi sia in tick (numero di cicli di clock a 40MHz) che in µs.

Coordinata Z Angolo di off nadir	391m 0,05rad	224m 0,8rad	6241m 0,05rad	3580m 0,8rad	8500m 0,05rad	4875m 0,8rad
Costo computazionale in Tick	24164	19916	384300	315860	523540	430084
Costo computazionale in µs	604,1	497,9	9'607,5	7'896,5	13'088,5	10'752,1

Figura 3.24 Tempi per la prima versione del calcolo dell'eco normalizzato

I tempi di calcolo, nei casi più sfavorevoli, supera il tempo di aggiornamento dei dati cinematici di 10ms. Si considera il tempo di 10ms come tempo di esecuzione massimo, quindi non è possibile utilizzare questa implementazione. Si devono eseguire delle ottimizzazioni per velocizzare il calcolo.

Logica utilizzata

La logica utilizzata per implementare il programma può essere estratta dallo XilinxLog. La prima versione del codice è sicuramente quella più lenta, perché non viene eseguita nessuna ottimizzazione, ma è quella in cui vengono utilizzati il minor numero di registri e di logica. I tempi di esecuzione registrati nelle prove sperimentali, tuttavia, escludono la possibilità di utilizzare questo codice nel VI FPGA. Vengono utilizzati:

- Slice Totali: 24,8% (3645 di 14720)
- Slice con Registri (SLICEM): 12,4% (7315 di 58880)
    - Usati come Flip Flop: 7315
- Slice con LUT (SLICEL): lice LUTs: 13,2% (7764 di 58880)
    - Usati come logica: 7538 di 58880
    - Usati come Memoria: 175 di 24320
        - Come RAM Dual Port: 132
        - Come Shift Register: 43
    - Usati come Route-Thru esclusivo: 51
- DSP48: 12,5% (80 di 640)
- Blocchi di RAM: 2,0% (5 di 244)

Inoltre viene riportato il report avanzato della sintesi HDL, dallo XilinxLog, per valutare le strutture basi impiegate. In questo caso sono stati utilizzati:

- RAM : 1
- Moltiplicatori : 35
- Addozionatori/Ssottrattori : 130
- Contatori : 23
- Accumulatori : 0
- Registri come Flip Flop : 8139
- Comparatori : 28
- Multiplexer : 1373
- Decoder : 2
- Macchine a Stati finiti : 15
- Xor : 40
- Register per le FSM : 7088
- Shift Register : 21

Un altro report importate è quello riguardante il timing. In questo report, viene evidenziata la frequenza massima, cui può lavorare il codice. Per ogni parte di codice, eseguita con un unico clock, viene specificata la frequenza massima cui può lavorare. In seguito è mostrato il report:

- 40 MHz Onboard Clock: 40,00 MHz (53,95 MHz massimo)
- 100 MHz Clock: 100,00 MHz (122,94 MHz massimo)
- TS_BusClk: 149,12 MHz (massimo)
- TS_SlowBusClk: 145,73 MHz (massimo)
- TS_DramClk200: 203,79 MHz (massimo)
- TS_DramClk200s90: 209,12 MHz (massimo)
- TS_IoRxClock: 333,33 MHz (massimo)
- TS_ClockGenXilinxV5x_TxDcm_TxHighSpeedClkDcm: 297,80 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxHighSpeedClkDcm: 450,25 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxLowSpeedClkDcm: 170,53 MHz (massimo)

In questo caso, il codice potrebbe essere eseguito più velocemente, fino ad una frequenza massima di 53,95MHz. Però, il miglioramento non porterebbe a sodisfare le specifiche che ci siamo imposti per l'esecuzione del codice.

Miglioramenti per il Calcolo dell'Eco Normalizzato

La modifica più rilevante, applicata l'implementazione del calcolo dell'eco riflesso, è l'utilizzo delle funzioni pipelined. Queste permettono di utilizzare un Single-Cycle Timed Loop per la generazione dei campioni. In questo modo, ogni campione verrà generato in un unico ciclo di clock. L'utilizzo delle funzioni pipelined, aggiunge una latenza iniziale al calcolo.

Pipeline

Le funzioni pipelined, utilizzate nella progettazione dell'EG, permettono di implementare delle funzioni che, normalmente, richiederebbero più cicli di clock per produrre un risultato valido in un Single-Cycle Timed Loop. Queste funzioni sono presenti nella palette Programming dei VI FPGA, selezionando le voci FPGA Math and Analysis e High Throughput Math. Le funzioni pipelined utilizzate per il calcolo dell'eco sono:

- somma, in Figura 3.25.
- prodotto, in Figura 3.26
- seno e coseno, in Figura 3.27
- esponenziale, in Figura 3.28
- reciproco, in Figura 3.29
- radice quadrata, in Figura 3.30

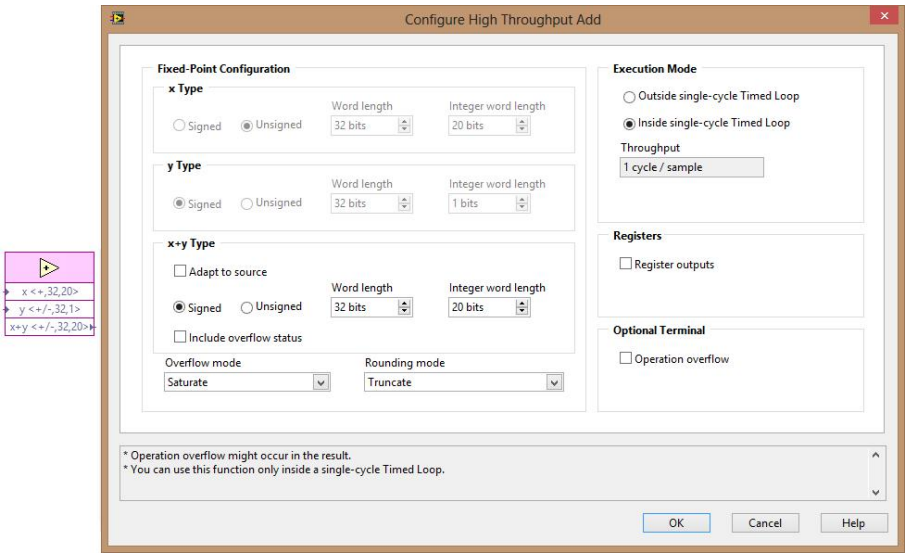

Figura 3.25 Funzione e pannello di configurazione per la somma Pipelined

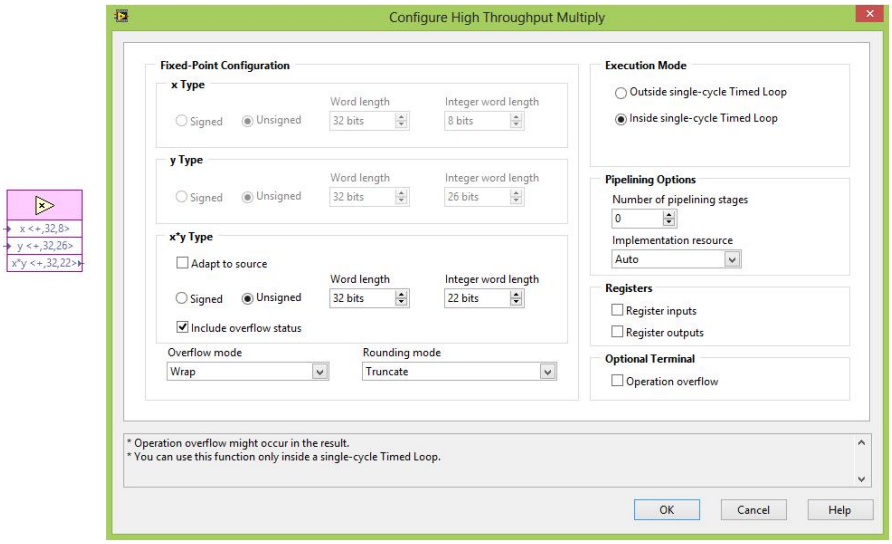

Figura 3.26 Funzione e pannello di configurazione per il prodotto Pipelined

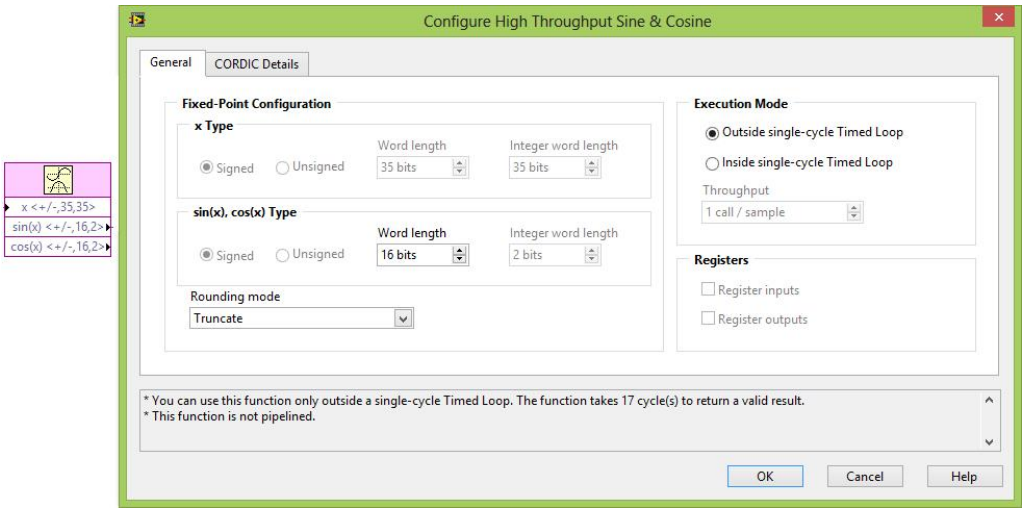

Figura 3.27 Funzione e pannello di configurazione per il calcolo del seno e del coseno Pipelined

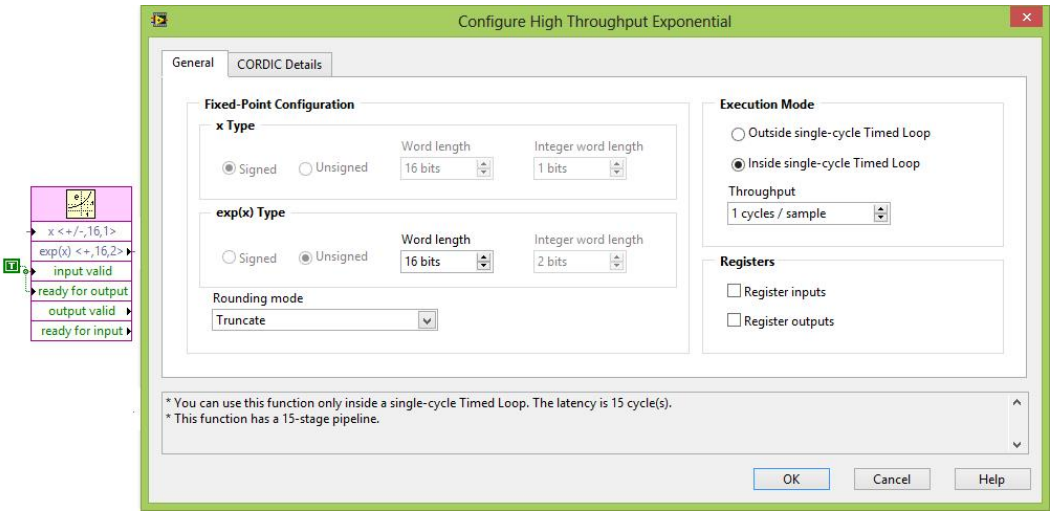

Figura 3.28 Funzione e pannello di configurazione per l'esponenziale Pipelined

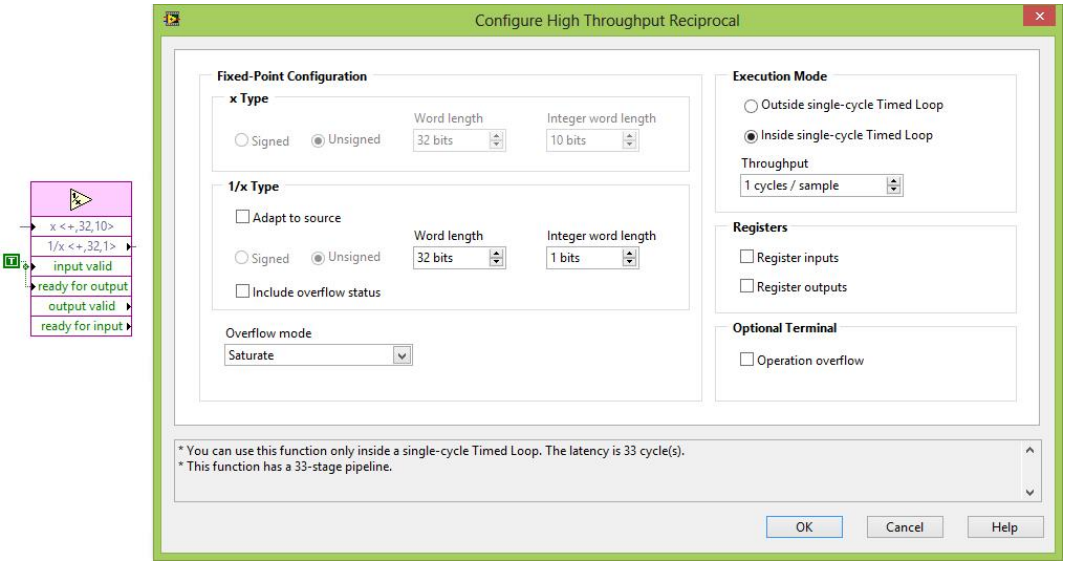

Figura 3.29 Funzione e pannello di configurazione per calcolare il reciproco Pipelined

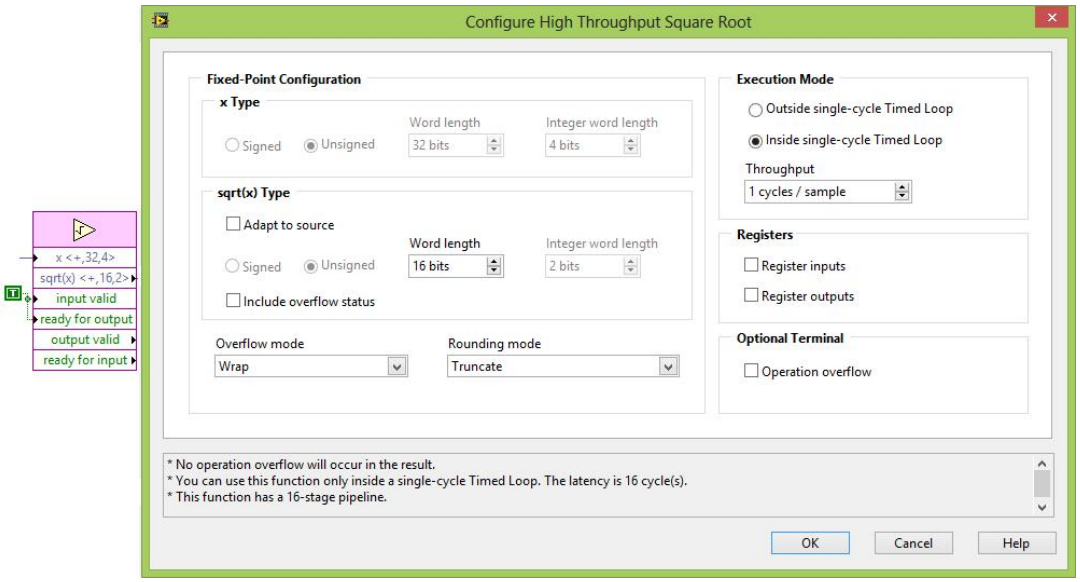

Figura 3.30 Funzione e pannello di configurazione per la radice quadrata Pipelined

Nelle immagini precedenti vengono mostrate, a sinistra le rappresentazioni, nel block diagram, delle funzioni pipelined e a destra, il loro pannello di configurazione. È possibile accedere al pannello di configurazione, semplicemente cliccando due volte sulla funzione. Nel pannello è possibile selezionare, nella parte sinistra, le dimensioni della word e il numero di bit da dedicare alla parte intera per gli ingressi e l'uscita. Per l'uscita, è possibile scegliere di visualizzare un elemento di segnalazione, dentro l'indicatore numerico della risorsa, per indicare se si è verificato un overflow. È possibile, inoltre, scegliere la modalità di Rounding, se si verifica un overflow nell'operazione. Nella parte bassa del pannello di configurazione, vengono mostrati dei messaggi come, per esempio, se è o non è possibile avere overflow nell'operazione. Nella parte destra del pannello, si può scegliere se eseguire la funzione all'interno o all'esterno di un Single-Cycle Timed Loop. Nel caso in cui si seleziona, come modalità di esecuzione, quella all'interno di un Timed Loop, è possibile scegliere il numero di cicli necessari per produrre un campione (si definiscono i numeri di cicli per campione). Nella parte inferiore del pannello, viene visualizzato un messaggio in cui viene descritta la latenza e il numero di stadi pipeline, necessari per implementare la funzione in un numero fissato di Single-Cycle Timed Loop. Inoltre, nella parte destra del pannello, è possibile scegliere se inserire dei registri di sincronizzazione, per i segnali d'ingresso e d'uscita.

Nel caso in cui le funzioni siano utilizzate in serie, all'interno di un Single-Cycle Timed Loop, è possibile che degli ingressi di una funzione arrivino in iterazioni differenti. Per esempio, nel caso in cui venga utilizzato come ingresso, di un moltiplicatore in versione High Throughput, le uscite di un addizionatore e di una funzione per il calcolo di un coseno, tutt'e due in versione High Throughput.

Le funzioni vengono configurate per la creazione di un campione al ciclo, quindi presenteranno una latenza iniziale. L'uscita dell'addizionatore sarà valida al primo ciclo ma l'uscita del coseno sarà valida solo dopo 17 cicli. Per introdurre la stessa latenza iniziale tra i due operatori, si può utilizzare la funzione mostrata in Figura 3.31. Questa funzione permette di introdurre una serie di registri e una linea di ritardo per i segnali, in modo da caricare i registri del pipeline correttamente.

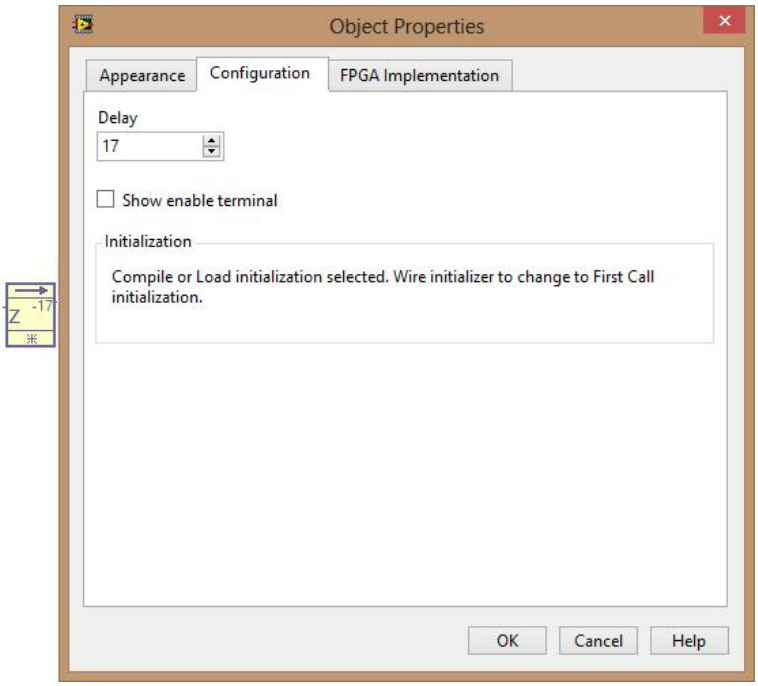

Figura 3.31 Funzione per inserire una linea di ritardo nel segnale

Implementazione delle funzioni

Le operazioni implementate nella versione pipeline, seguono lo stesso approccio utilizzato nella prima versione. Le operazioni che, nell'equazione di G(ε), non dipendono dall'istante di tempo considerato, vengono elaborate all'esterno del Single-Cycle Timed Loop. Le costanti fisse per il calcolo di ogni eco, come la dimensione dell'impulso riflesso, il Cos(ξ), il $Cos^2(\xi)$, il $Sen^2(\xi)$, il Sen(2ξ) e il t_{ric}-t_0 saranno calcolate solo una volta per ogni eco. Una volta valutati questi parametri, si prosegue con il calcolo dei campioni all'interno del Single-Cycle Timed Loop. Il codice per il calcolo dell'eco riflesso normalizzato è mostrato nella Figura 3.32. Nel primo e nell'ultimo frame sono presenti le funzioni Tick Count, che permettono di valutare il tempo necessario per il calcolo di un intero eco normalizzato.

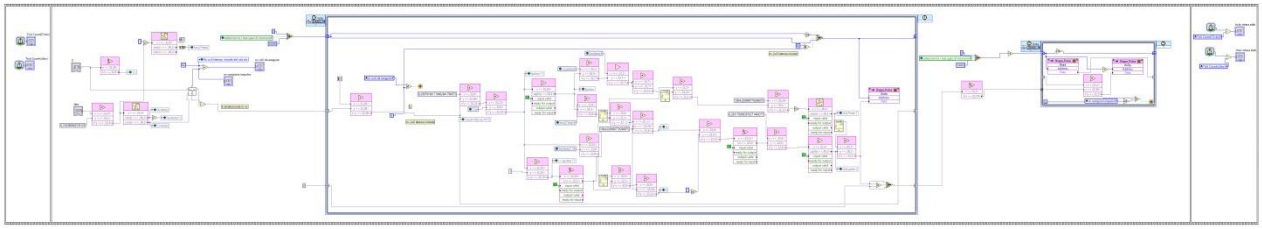

Figura 3.32 Seconda versione del codice per il calcolo dell'eco normalizzato

Per calcolare e produrre un segnale valido all'uscita della catena dei registri di pipeline, da implementare nel Single-Cycle Timed Loop, è necessario inserire degli elementi di ritardo nella propagazione del segnale. Il primo passo, per lo studio dell'implementazione del calcolo, è quello di rappresentare graficamente le operazioni che si devono implementare. Nella Figura 3.33, vengono rappresentate le operazioni con, in rosso sopra ogni funzione, la latenza iniziale.

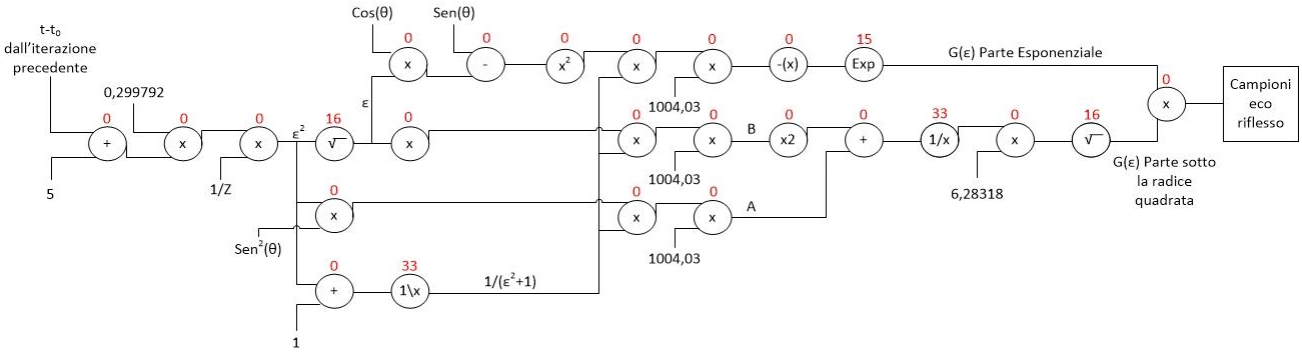

Figura 3.33 Operazioni, con relativi tempi di latenza, da eseguire in un'iterazione del Single-Cycle Timed Loop per il calcolo dell'eco

Nella Figura 3.34, vengono rappresentati i tempi di arrivo dei dati. Si assume che gli ingressi esterni siano tutti disponibili nell'istante iniziale. Nel caso in cui i dati d'ingresso arrivano in una funzione, in due istanti differenti, il risultato in uscita sarà errato, perché generato da fattori provenienti da istanti diversi.

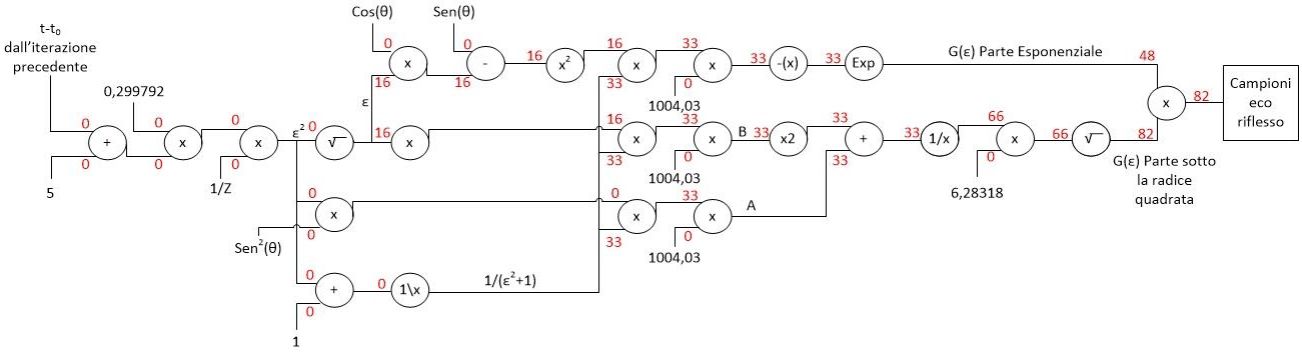

Figura 3.34 Tempi di arrivo dei dati per ogni operazione

Per implementare il calcolo sono necessari, quindi, degli elementi di ritardo, per ottenere un risultato corretto. In Figura 3.35, gli elementi di ritardo sono rappresentati con dei quadrati. All'interno di ogni elemento di ritardo, viene indicato il valore del ritardo necessario per ottenere un'uscita corretta. Nella Figura 3.35 sono rappresentati i nuovi tempi di arrivo.

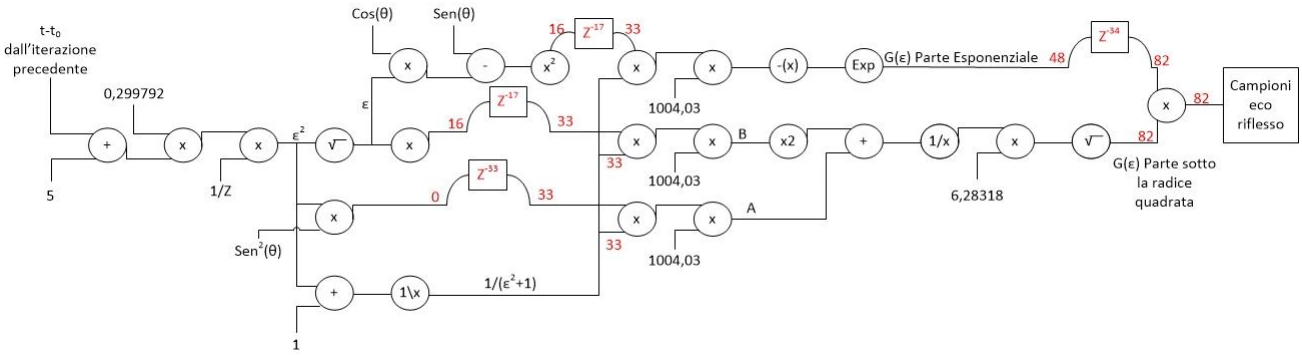

Figura 3.35 Versione corretta dell'algoritmo, in cui sono stati inseriti gli elementi di ritardo

Con l'inserimento dei ritardi, l'arrivo degli ingressi per ogni funzione, avverrà nello stesso istante. Il codice corretto è mostrato nella Figura 3.32, in cui sono stati già inseriti i blocchi di ritardo.

Costo computazionale

Utilizzando questi miglioramenti, i tempi di esecuzione diminuiscono sensibilmente. Nella Figura 3.36 vengono mostrati i tempi di esecuzione, necessari per la seconda versione del calcolo dell'eco normalizzato. I tempi di esecuzione vengono espressi sia in tick (numero di cicli di clock a 40MHz) che in µs.

Coordinata Z	391m	224m	6241m	3580m	8500m	4875m
Angolo di off nadir	0,05rad	0,8rad	0,05rad	0,8rad	0,05rad	0,8rad
Costo computazionale in Tick	601	544	5484	4561	7370	6110
Costo computazionale in µs	15,025	13,6	137,1	114,025	184,25	152,75

Figura 3.36 Tempi per la seconda versione del calcolo dell'eco normalizzato

Nela Figura 3.37, viene mostrato il confronto tra le due versioni implementate.

Coordinata Z	391m	224m	6241m	3580m	8500m	4875m
Angolo di off nadir	0,05rad	0,8rad	0,05rad	0,8rad	0,05rad	0,8rad
Costo computazionale della prima versione in µs	604,1	497,9	9'607,5	7'896,5	13'088,5	10'752,1
Costo computazionale della seconda versione in µs	15,025	13,6	137,1	114,025	184,25	152,75
Miglioramento percentuale	97,5%	97,2%	98,6%	98,5%	98,6%	98,6%

Figura 3.37 Confronto tra i tempi di esecuzione, per le due versioni del calcolo dell'eco normalizzato

Il miglioramento registrato, calcolato come la percentuale del tempo risparmiato per il calcolo, rispetto al costo computazionale della prima versione, parla da solo. Questa versione può essere utilizzata efficacemente. Il codice sarà eseguito ogni 1ms, ossia un tempo minore rispetto a quello di aggiornamento dei dati cinematici, provenienti dal simulatore dell'interfaccia con il modulo EDM. Il codice potrebbe essere eseguito anche ogni 500µs, fino ad un minimo di 200µs, senza l'aumento del costo computazionale.

Logica utilizzata

La logica utilizzata per implementare il programma può essere estratta dallo XilinxLog. Questa versione del codice è più veloce, in conseguenza sarà utilizzato un numero maggiore di registri e di logica. Sono utilizzati:

- Slice Totali: 39,7% (5846 di 14720)

- Slice con Registri (SLICEM): 23,5% (13836 di 58880)
    - Usati come Flip Flop: 13836
- Slice con LUT (SLICEL): 21,6% (12711 di 58880)
    - Usati come logica: 12284 di 58880
    - Usati come Memoria: 365 di 24320
        - Come RAM Dual Port: 132
        - Come Shift Register: 233
    - Usati come Route-Thru esclusivo: 62
- DSP48: 9,2% (59 di 640)
- Blocchi di RAM: 38,9% (95 di 244)

Confrontando questi valori con quelli rilevati nella prima versione, si nota un aumento delle risorse utilizzate. Quest'aumento era facilmente prevedibile poiché, in questa versione, sono utilizzate delle elaborazioni aritmetiche pipelined. La percentuale delle slice totali utilizzate passa dal 24,8% al 39,7%, con un aumento delle SLICEM (dal 12,4% al 23,5%) e delle SLICEL (dal 13,2% al 21,6%). Questo è imputabile all'utilizzo sia di funzioni pipelined, sia della struttura Single-Cycle Timed Loop. Diminuiscono i DSP48E (dal 12,5% al 9,2%) utilizzati. L'aumento più rilevante si può osservare per i blocchi di RAM, che passano dal 2% al 38,9%.

È riportato, inoltre, il report avanzato della sintesi HDL dallo XilinxLog, per valutare le strutture basi impiegate. In questo caso, sono stati utilizzati:

- RAM : 3
- Moltiplicatori : 4
- Addizionatori/Sottrattori : 433
- Contatori : 24
- Accumulatori : 0
- Registri come Flip Flop : 17200
- Comparatori : 356
- Multiplexer : 2386
- Decoder : 2
- Macchine a Stati finiti : 16
- Xor : 175
- Registri per le FSM : 9941
- Shift Register : 219

La variazione più rilevante, rispetto alla prima versione, si osserva per l'utilizzo dei registri come Flip Flop, che passano da 8139 a 17200. L'aumento è imputabile all'utilizzo delle funzioni aritmetiche pipelined.

Un altro report importate è quello riguardante il timing. In questo report, viene evidenziata la frequenza massima, cui può lavorare il codice. Per ogni parte di codice, eseguita con un unico clock, viene specificata la frequenza massima cui può lavorare. In seguito è mostrato il report:

- 40 MHz Onboard Clock: 40,00 MHz (76,80 MHz massimo)

- 100 MHz Clock: 100,00 MHz (120,02 MHz massimo)
- 25MHz: 25,00 MHz (27,60 MHz massimo)
- TS_BusClk: 140,49 MHz (massimo)
- TS_SlowBusClk: 166,36 MHz (massimo)
- TS_DramClk200: 202,51 MHz (massimo)
- TS_DramClk200s90: 268,02 MHz (massimo)
- TS_IoRxClock: 333,33 MHz (massimo)
- TS_ClockGenXilinxV5x_TxDcm_TxHighSpeedClkDcm: 299,58 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxHighSpeedClkDcm: 450,25 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxLowSpeedClkDcm: 158,08 MHz (massimo)

In questo caso, è stata utilizzata la compilazione per conoscere la frequenza massima di esecuzione del Single-Cycle Timed Loop, utilizzato per il calcolo dei campioni dell'impulso riflesso e la successiva normalizzazione. La frequenza di clock sintetizzabile, su FPGA, più vicina a quella massima di funzionamento del codice (27,6MHz), è di 25MHz. Questa frequenza non può essere determinata prima della compilazione, perché non si conoscono le risorse utilizzate sull'FPGA. Solo dopo l'operazione di Place and Route della compilazione, è possibile conoscere la frequenza massima di esecuzione.

Implementazione dell'acquisizione dei dati su FPGA

Il Vi, eseguito sull'FPGA per l'acquisizione, effettua il controllo e la temporizzazione del sistema, l'acquisizione dall'ingresso analogico e la scrittura dei dati sulla coda FIFO OUT DATI. Inoltre, esegue una valutazione del PRI, per acquisire un numero corretto di campioni del segnale generato, evitando di eseguire uno streaming di elementi continuo e il trasferimento di dati ausiliari, per una corretta interpretazione dei dati acquisiti. L'EG deve eseguire un'acquisizione di una parte del segnale d'ingresso, dipendente dal periodo di PRI inviato dl radar. Dalle specifiche dell'EG, si possono trovare tre valori della PRI corrispondenti a tre posizioni limiti del radar. La posizione limite viene definita con i valori dello Slant Range, ossia della quota del radar. I valori sono mostrati nella Figura 3.38.

Periodo di PRI inviato dal radar [µs]	Numero di campioni da acquisire tra due PRI	Tempo di acquisizione effettiva con f_{ck} di 200MHz [µs]	Percentuale del tempo d'acquisizione, rispetto al periodo di PRI	Velocità di trasferimento dall'FPGA all'Host richiesta [MB/s]
10	70	0,35	3,5%	28
46	250	1,25	2,7%	21
88	750	3,75	4,2%	34

Figura 3.38 Possibili valori di PRI e numero di campioni da acquisire a 200MHz

Nella prima parte del programma, come mostrato nello schema a blocchi in Figura 3.39, si va a configurare l'FPGA. In questa fase, si attende l'inizializzazione del Modulo Adapter NI 5761; si predispone l'FPGA per il possibile utilizzo di un clock esterno; si inizializza la DRAM, presente nella FlexRIO PXIe-7965R, eliminando eventuali dati presenti da acquisizioni precedenti; si esegue la sincronizzazione con il clock a 10 MHz, utilizzato per sincronizzare l'intero sistema. In seguito,

vengono eseguiti i cicli per la generazione dei segnali di temporizzazione e di controllo, per l'acquisizione dei dati e per il trasferimento dei dati all'Host. L'operazione di acquisizione viene controllata con un segnale di abilitazione, dall'unità di controllo, e dalla parte del codice che permette la scelta della finestra di acquisizione. La DRAM, presente nella PXIe-7965R, permette di bufferizzare il trasferimento. Quindi, i campioni acquisiti verranno, in un primo momento, memorizzati sulla DRAM, e poi inviati al processore, per il loro salvataggio su disco.

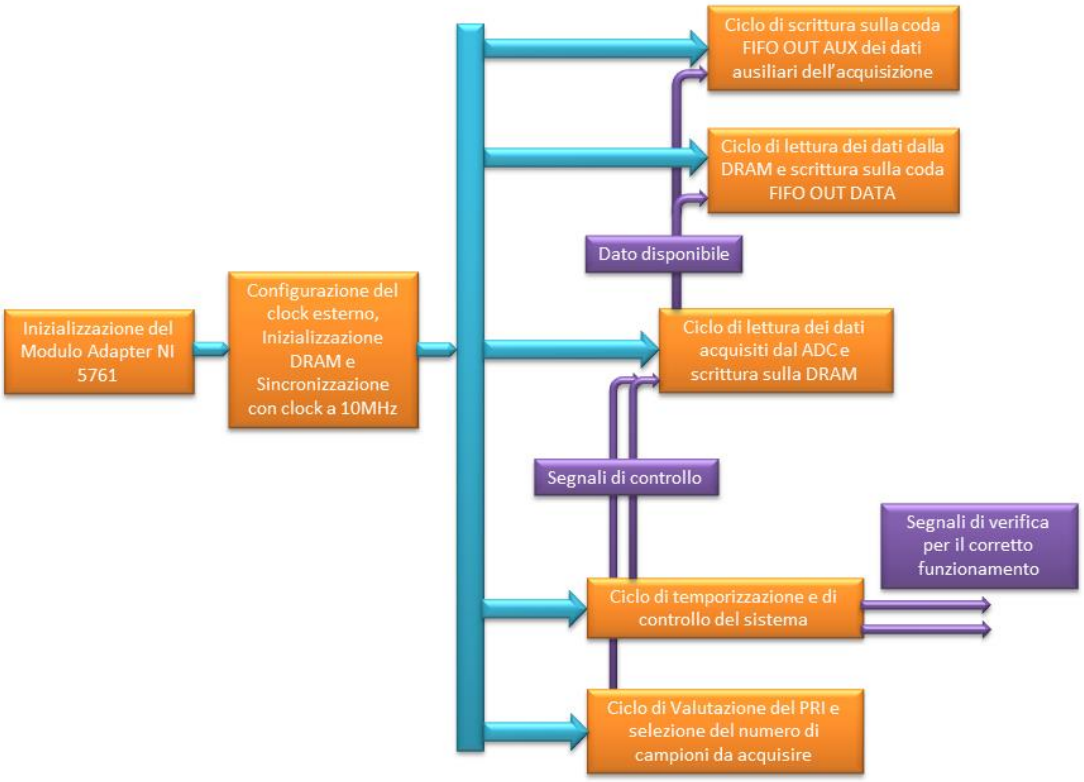

Figura 3.39 Schema a blocchi del programma eseguito su FPGA per l'acquisizione

Prima di scrivere il codice vero e proprio, si devono configurare, all'interno del progetto, tutte le unità hardware utilizzate.

Quando viene aggiunto il target PXIe-7965R ad un progetto, si deve selezionare il modulo Adapter utilizzato. Premendo il pulsante destro del mouse su IO Module, presente nel sottomenù del PXIe-7965R, selezionando la voce Properties, è possibile scegliere il modulo adapter corretto. Nel nostro caso, sarà selezionato il modulo NI 5761. Labview carica, in automatico, la CLIP di default, utilizzata per visualizzare i nomi dei terminali, all'interno degli IO Node. La CLIP utilizzata è la Multiple Sample CLIP, che permette di acquisire due campioni per ogni ciclo di clock. In seguito, vengono aggiunti i clock, utilizzati nel sistema, a 40MHz, a 100MHz e il PXI10, il clock a 10 MHz al progetto. Il clock a 40MHz sarà utilizzato nelle fasi di configurazione iniziale e per il trasferimento dei campioni dalla DRAM alla FIFO OUT DATA. Quello a 100MHz sarà impiegato per l'unità di controllo, per l'acquisizione e la valutazione del periodo di PRI. Infine, saranno aggiunte al progetto le code FIFO OUT, necessarie per il trasferimento dei dati, e la memoria DRAM, utilizzata come una coda FIFO.

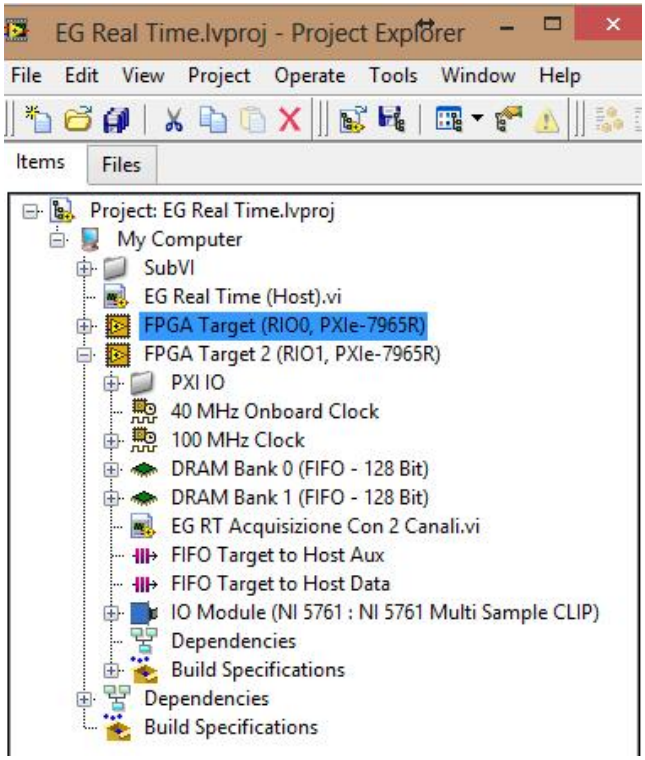

Figura 3.40 Il progetto, i SubVI e le unità hardware utilizzate su FPGA per l'acquisizione

Per la creazione di una coda FIFO, si deve premere il pulsante destro del mouse sul target, seleziona new e poi FIFO. Nella finestra di configurazione, appena aperta, sarà indicato il nome della FIFO, il tipo di coda da creare, le modalità di implementazione su FPGA, il tipo di dati processati e il numero di elementi massimi, contenuti nella FIFO. Per l'acquisizione, sono utilizzate due code FIFO DMA, la FIFO OUT DATA e la FIFO OUT AUX, di tipo Target to Host–DMA con 32767 elementi di tipo Intero a 16bit.

Una volta configurato il target, si può passare alla programmazione del VI FPGA. Il programma è stato diviso in varie macro aree, individuate con una struttura Flat Sequence. Questa permette di eseguire le varie porzioni di codice, in maniera sequenziale.

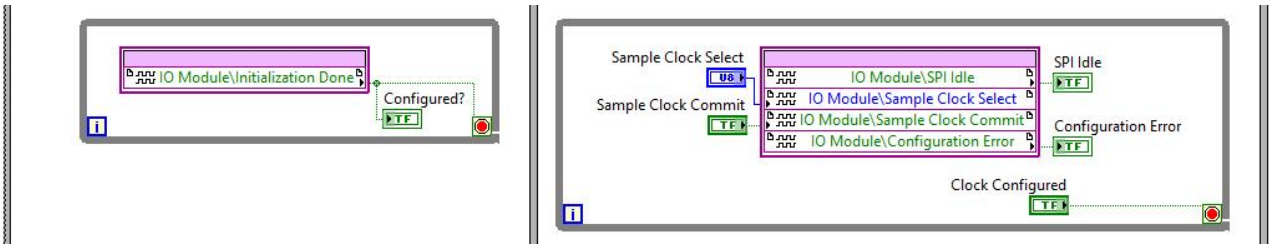

Figura 3.41 Codice per il controllo dell'inizializzazione del modulo NI 5761, della configurazione del clock e completamento delle operazioni iniziali

Il primo blocco, in Figura 3.41, permette di confermare che il modulo Adapter I/O sia stato inizializzato correttamente, con il valore di Inizializzation Done. Questo passaggio deve essere eseguito, ogni volta che un VI deve ottenere i privilegi per l'accesso esclusivo al modulo Adapter. Il secondo blocco, in Figura 3.41, permette la configurazione del clock. La selezione del clock viene effettuata con la variabile Sample Clock Select, impostata dal VI Host con una variabile enumerativa. Se la variabile assume i valori 0 e 2, non si deve attende che il PLL sia agganciato. Se

assume i valori 1 o 3, il clock non sarà considerato configurato, fino a che il PLL non risulti agganciato al clock di riferimento. In questa sezione si attende sempre che SPI Idle sia ad un valore logico positivo, perché il bus SPI permette di configurate i registri presenti nel chip nell'ADC. Il programma host invierà al VI FPGA, dopo la ricezione positiva dell'SPI Idle, il segnale Clock Configured. Questo segnale permetterà, al VI FPGA, di continuare la sua esecuzione.

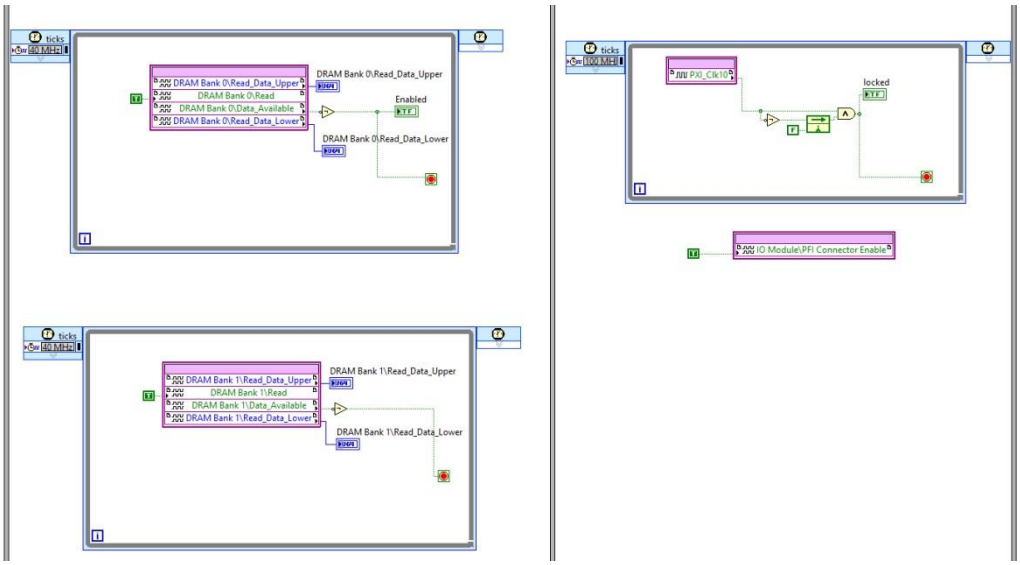

Figura 3.42 Codice per l'inizializzazione della DRAM e per la sincronizzazione con il clock a 10MHz

Il blocco di codice successivo, mostrato nel frame a sinistra in Figura 3.42, permette di eseguire una lettura di dati dalla DRAM. La lettura prosegue fin quando sono presenti dati nella DRAM, ossia fin quando rimane alto il segnale Data_Available. Questa variabile, se assume il valore alto, segnala la presenza di dati nella DRAM. Essendo una coda FIFO, quando un dato è letto dalla DRAM, viene immediatamente cancellato dalla coda. La DRAM, presente nel PXIe-7965R, è composta da due banchi da 256MB. Quindi, sono necessari due cicli While paralleli per svuotarli tutt'e due. Una volta completato lo svuotamento della DRAM, si può eseguire il secondo blocco in Figura 3.42. Questa porzione di codice permette di sincronizzare il clock a 100MHz, generato sull'FPGA, con il clock a 10MHz presente nel backplane. La sincronizzazione viene eseguita con un riconoscitore di fronte di salita, applicato al clock a 10 MHz, e un ciclo While, che ferma l'esecuzione del codice, fin quando non viene rilevato un fronte di salita, del 10MHz, in corrispondenza di un fronte di salita del 100MHz.

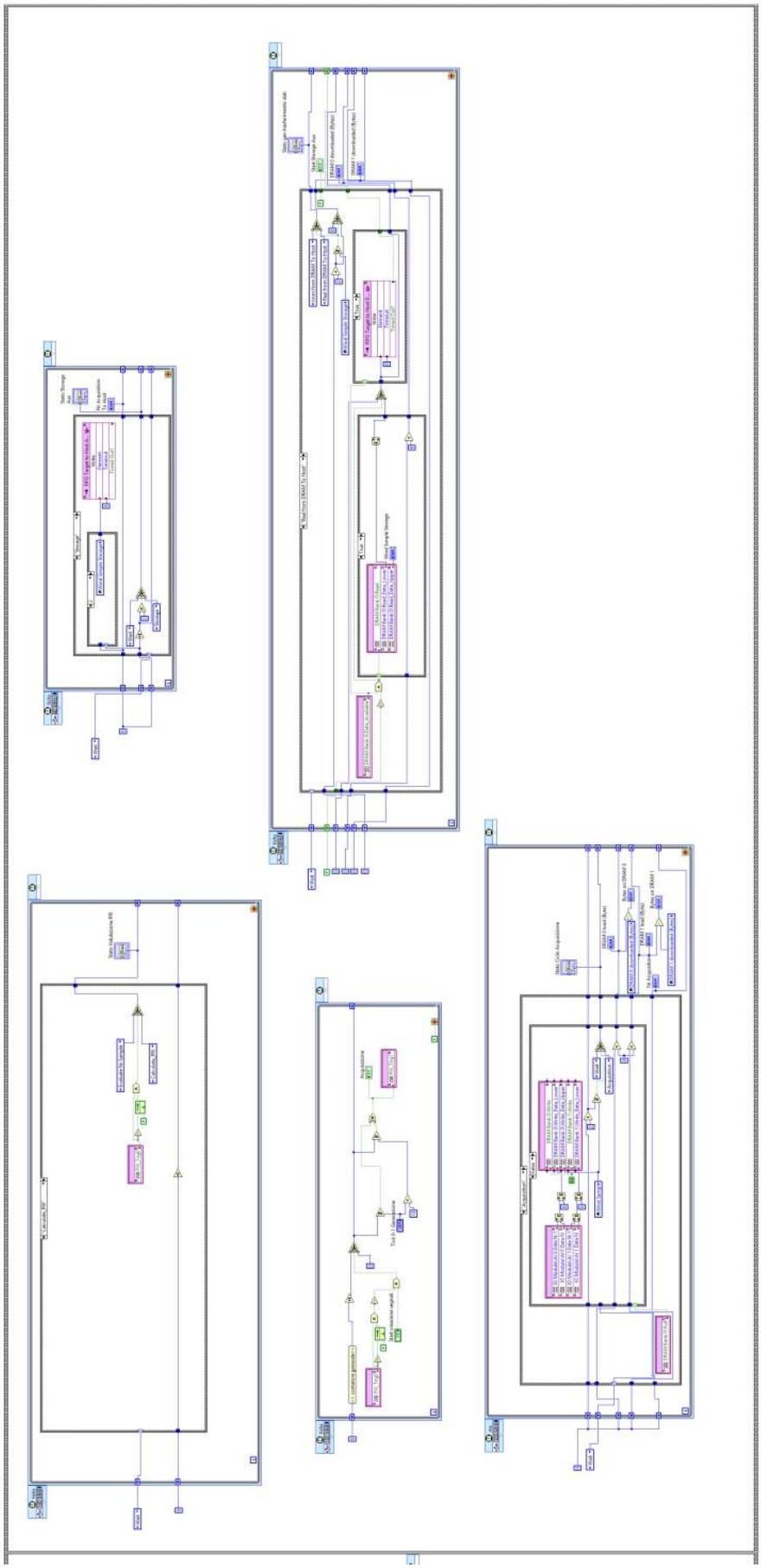

Figura 3.43 Codice, su FPGA, per l’acquisizione dei campioni

Il blocco successivo, mostrato in Figura 3.43, è il vero programma che, eseguito all'interno dell'FPGA, permette la temporizzazione e l'acquisizione dei dati.
In questa sequenza, sono presenti cinque cicli Timed Loop. I cinque cicli permettono di:

- valutare il periodo di PRI e selezionare l'ampiezza della finestra di acquisizione (in alto a sinistra)
- temporizzare la struttura, generando un segnale, chiamato Acquisizione, per abilitare il salvataggio dei campioni nella DRAM. (al centro nella colonna di sinistra)
- acquisire i due segnali e salvare i campioni sulla DRAM (in basso a sinistra)
- inviare i dati dalla DRAM al VI Host per la memorizzazione (al centro a destra)
- inviare dei dati ausiliari al VI Host per una corretta interpretazione dei dati (in alto a destra)

Questi cinque cicli devono essere sempre eseguiti. Per questo, viene collegata una costante, con valore booleano falso, alla condizione di interruzione dei cicli.

Il blocco che si occupa della temporizzazione è stato implementato con l'utilizzo di alcuni Shift Register, di sommatori, di comparatori e di logica booleana. Il contatore generale utilizza uno Shift Register, inizializzato a zero. Questo contatore è incrementato se il segnale Start Creazione Segnali è abilitato, e se non viene rilevato un fronte di salita sul segnale di PRI. Nel caso contrario, il contatore viene portato a zero, mantenendo fermo il suo valore. Il PRI è collegato all'interfaccia digitale del NI 5781 (collegato all'FPGA utilizzata per la generazione) e inviato attraverso una linea di trigger, PXI_Trig0 presente nel blackplane, all'FPGA per l'acquisizione. Utilizzando il contatore e conoscendo il periodo di clock, è possibile calcolare dei ritardi rispetto ad un fronte di salita del PRI, per portare ad un livello logico alto o basso i segnali di controllo del sistema. Nell'unità di controllo viene generato il segnale Acquisizione che permette di avviare l'acquisizione. Il primo comparatore controlla se il contatore sia maggiore del ritardo e, nel caso positivo, porta ad un livello logico alto il segnale. Il secondo comparatore e una porta Xor riportano ad un livello logico basso, il segnale Acquisizione, se viene superato un valore limite. L'uscita della Xor viene collegata ad una linea di trigger (PXI_Trig3), per inviare il segnale alle porte digitali dell'NI 5781 e, per esempio, visualizzarlo su di un oscilloscopio. Utilizzando un clock a 100MHz, il ritardo dal fronte di salita del PRI può essere solo un multiplo del periodo di clock, ossia 10ns.

Gli altri quattro Single-Cycle Timed Loop sono implementati come delle macchine a stati finiti. Questa struttura può essere realizzata, grazie alla presenza del Case Structure (per la selezione degli stati) e di variabili enumerative (per associare un'etichetta ad un valore intero). Nelle variabili enumerative sono definiti gli stati della FSM e, all'interno del case structure, viene descritto il codice da eseguite in ogni stato.

Il codice, per valutare il periodo di PRI e per selezionare l'ampiezza della finestra di acquisizione, è descritto come una FSM a tre stati. In quello iniziale si attende la prima transizione basso-alto del PRI, nel secondo si avvia il contatore e si controlla se viene ricevuto un nuovo fronte di salita sul PRI. Se viene ricevuto un nuovo fronte di salita si passa nel terzo stato, in cui viene letto il periodo di PRI e selezionato il numero di campioni da acquisire.

Il codice, per acquisire i due segnali e salvare i campioni sulla DRAM, è stato implementato come una FSM con due stati. Il primo stato, denominato Wait, permette di attendere che il segnale Acquisizione venga portato alto. Una volta rilevato un fronte di salita del segnale, si passa al

secondo stato, in cui si esegue l'acquisizione dei due canali. In questo stato, è eseguita la scrittura dei dati sulla DRAM. I due banchi di RAM permettono la memorizzazione, dei dati provenienti dai due canali, in due aree di memoria separate.

Il codice, per inviare i dati dalla DRAM al VI Host per la memorizzazione, è stato implementato come una FSM a tre stati. Il primo stato è di Wait, in cui si attende che i dati siano disponibili per la lettura dalla DRAM. La variabile Data_Available, la quale indica se sono presenti dei dati nella DRAM per la lettura, deve trovarsi in un livello logico alto, per passare nello stato successivo. Il codice del secondo stato permette il trasferimento dei dati dalla DRAM alla RAM del processore. In questo caso, si esegue la lettura dei dati dalla DRAM se la coda FIFO OUT DATA è pronta per il trasferimento di nuovi dati. Nel caso in cui occorre una condizione d'errore, indicata dalla FIFO con il terminale Timed Out, non viene letto un nuovo dato dalla DRAM, ma viene ritrasferito il dato letto nella precedente iterazione. Una volta processati tutti i campioni di un canale, si passa al terzo stato in cui sono trasferiti i campioni dell'altro canale. Una volta completato il trasferimento, la FSM ritorna allo stato di Wait.

L'ultima porzione di codice riguarda il trasferimento di dati ausiliari per la corretta interpretazione dei dati. Una FSM a due stati è stata utilizzata per descrivere il codice. Il primo stato attende che sia avviato il trasferimento dei campioni sulla coda FIFO OUT DATA, per avviare il trasferimento dei dati ausiliari sulla coda FIFO OUT AUX. Vengono trasferiti come valori ausiliari, un valore costante (32000), il numero dell'acquisizione e il numero dei campioni acquisiti.

Implementazione della generazione dei dati su FPGA

Il VI, eseguito sull'FPGA per la generazione, effettua il controllo e la temporizzazione del sistema, il calcolo dell'eco riflesso normalizzato, il calcolo delle componenti I&Q e l'invio dei campioni all'AWG per la generazione. Inoltre, esegue una valutazione del PRI (utilizzato nel calcolo dell'aggiornamento dei dati cinematici) e la simulazione dei dati cinematici ricevuti dal modulo EDM. Nella prima parte del programma, come mostrato nello schema a blocchi in Figura 3.44, si va a configurare l'FPGA. In questa fase, si attende l'inizializzazione del Modulo Adapter NI 5781; si predispone l'FPGA per il possibile utilizzo di un clock esterno; si inizializza la DRAM, presente nella FlexRIO PXIe-7965R, eliminando eventuali dati presenti da acquisizioni precedenti; si esegue la sincronizzazione con il clock a 10 MHz, utilizzato per sincronizzare l'intero sistema. In seguito, vengono eseguiti i cicli per la generazione dei segnali di temporizzazione e di controllo; per il calcolo dell'impulso normalizzato riflesso dalla superficie marziana; per il calcolo delle componenti I&Q per la generazione; per il trasferimento dei campioni all'AWG per la generazione. Le operazioni di calcolo dell'impulso normalizzato, riflesso dalla superficie marziana, e delle componenti I&Q, sono state presentate ampliamente nei paragrafi precedenti. I campioni dell'impulso normalizzato vengono memorizzati in una memoria, Shape_Pulse, implementati in blocchi di memoria nell'FPGA. I campioni, rappresentanti le componenti I&Q del segnale, vengono inviati al ciclo per il trasferimento all'AWG con una coda FIFO chiamata Dati I&Q. Questa coda è definita come una coda FIFO Target-Scoped a 64bit, implementata nei blocchi di memoria dell'FPGA.

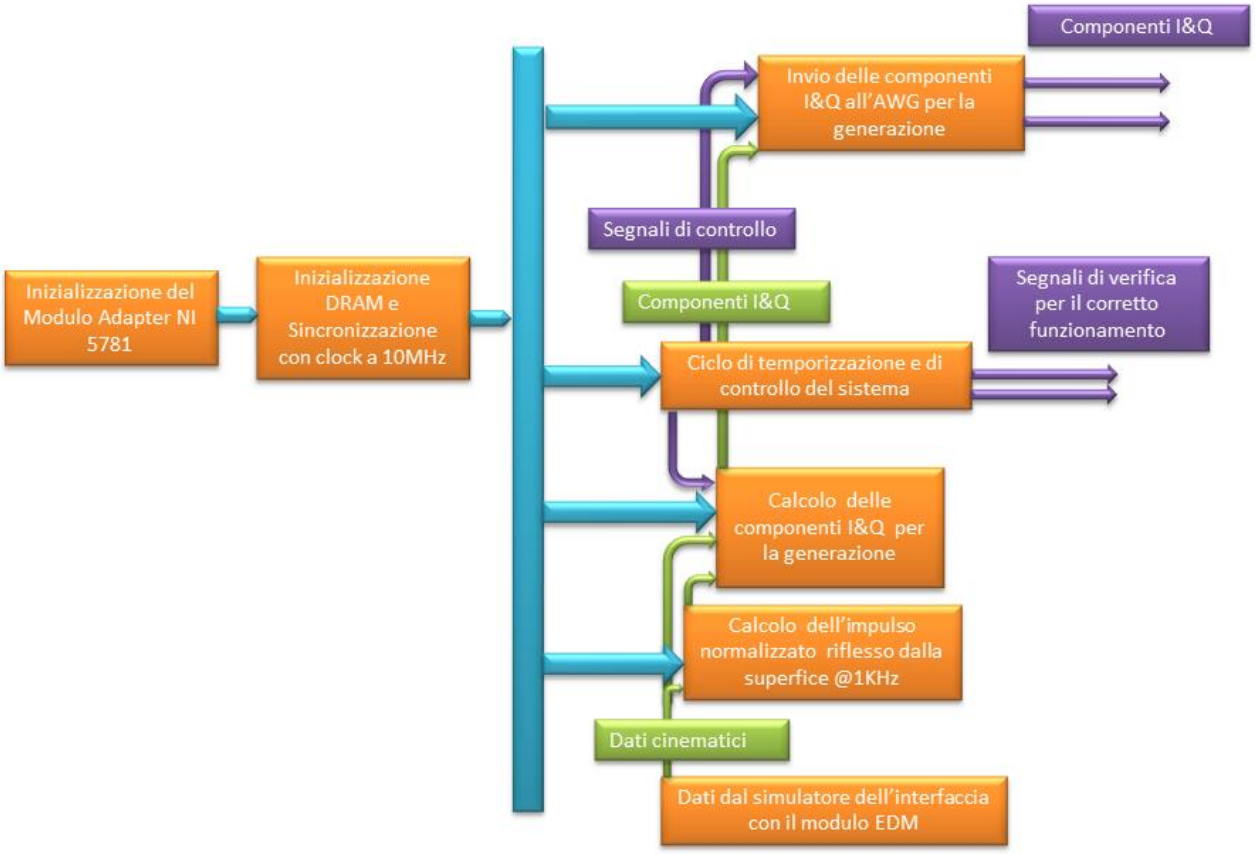

Figura 3.44 Schema a blocchi del programma eseguito su FPGA per la generazione

Come prima operazione si devono configurare, all'interno del progetto, tutte le unità hardware utilizzate.

Quando viene aggiunto il target PXIe-7965R ad un progetto, si deve selezionare il modulo Adapter utilizzato. Nel nostro caso, sarà selezionato il modulo NI 5761. Labview caricherà, in automatico, la CLIP di default, utilizzata per visualizzare i nomi dei terminali, all'interno degli IO Node. In seguito, vengono aggiunti i clock, utilizzati nel sistema, a 40MHz, a 100MHz, a 25MHz (derivato con un divisore di frequenza dal clock a 100MHz) e il PXI10, il clock a 10 MHz al progetto. Il clock a 40MHz sarà utilizzato nelle fasi di configurazione iniziale, nel calcolo per l'aggiornamento dei dati cinematici e, nel ciclo while, utilizzato per simulare i dati provenienti dall'interfaccia con il modulo EDM. Quello a 100MHz sarà impiegato per l'unità di controllo, per l'invio dei campioni all'AWG e la valutazione del periodo di PRI. Quello a 25MHz sarà utilizzato solo per il calcolo dell'eco riflesso, per la sua normalizzazione e per il calcolo delle componenti I&Q del segnale. In seguito, viene aggiunta al progetto la coda FIFO Target-Scoped "Dati I&Q", necessarie per il trasferimento dei campioni all'interno del programma FPGA; la memoria Shape_Pulse, per la memorizzazione dell'eco normalizzato; la coda Peer To Peer tra l'FPGA e l'AWG, chiamata P2P FPGA To AWG, per il trasferimento diretto dei dati senza l'uso del processore. Il progetto realizzato è mostrato nella Figura 3.45.

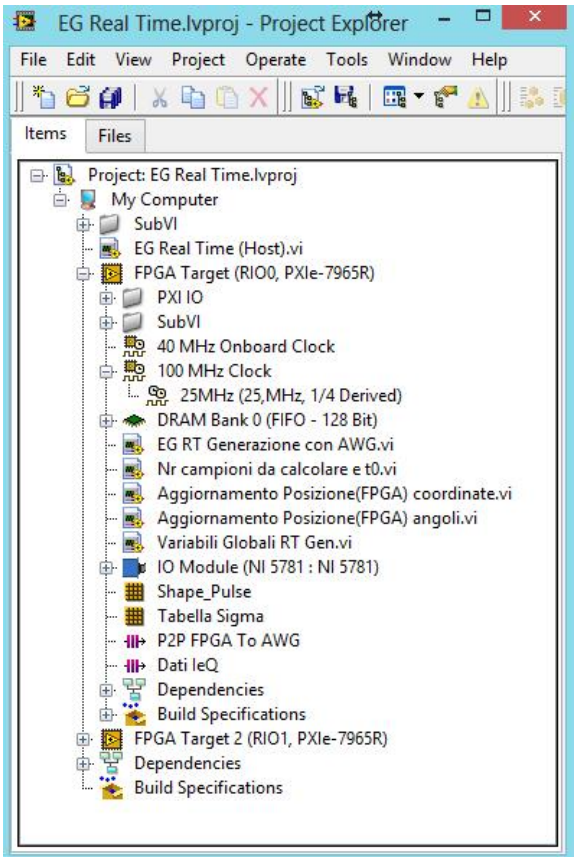

Figura 3.45 Il progetto, i SubVI e le unità hardware utilizzate su FPGA per la generazione

Una volta configurato il target, si può passare alla programmazione dei VI FPGA. Il programma è stato diviso in varie macro aree, individuate con la struttura Flat Sequence. Questa permette di eseguire le varie porzioni di codice in maniera sequenziale.

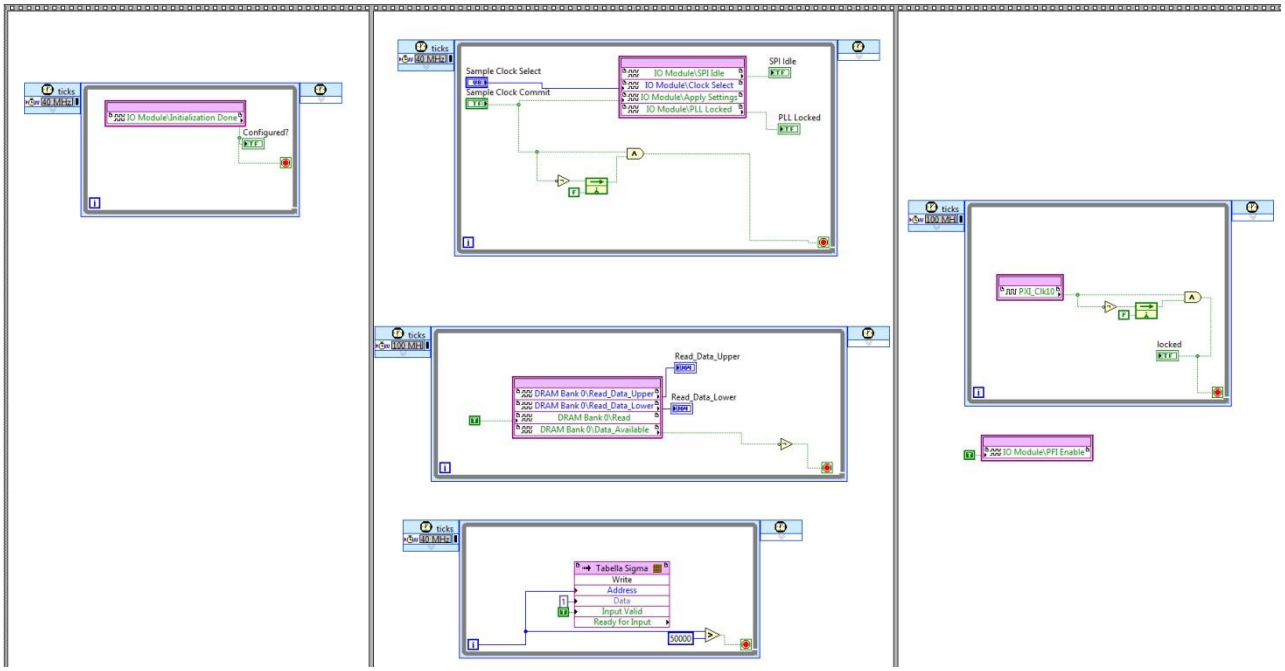

Figura 3.46 Codice, su FPGA, per l'inizializzazione dell'NI5781, della DRAM e per la sincronizzazione con il clock a 10MHz

Il primo blocco, in Figura 3.46, permette di confermare che il modulo Adapter I/O sia stato inizializzato correttamente, con il valore di Inizializzation Done. Questo passaggio deve essere eseguito, ogni volta che un VI deve ottenere i privilegi per l'accesso esclusivo al modulo Adapter.

Il secondo blocco, in Figura 3.46, è composto da tre Single-Cycle Timed Loop. Il primo permette la configurazione del clock, il secondo, l'inizializzazione dalla DRAM. Infine, il terzo loop permette di scrivere dei valori di default, nella locazione di memoria per la variabile σ, utilizzata il calcolo della potenza reale del segnale. Una volta completate queste operazioni, si può eseguire il terzo blocco in Figura 3.46.

Questa porzione di codice permette di sincronizzare il clock a 100MHz, generato sull'FPGA, con il clock a 10MHz, presente nel backplane La sincronizzazione viene eseguita con un riconoscitore di fronte di salita, applicato al clock a 10 MHz, e un ciclo While, che ferma l'esecuzione del codice, fin quando non viene rilevato un fronte di salita, del 10MHz, in corrispondenza di un fronte di salita del 100MHz..

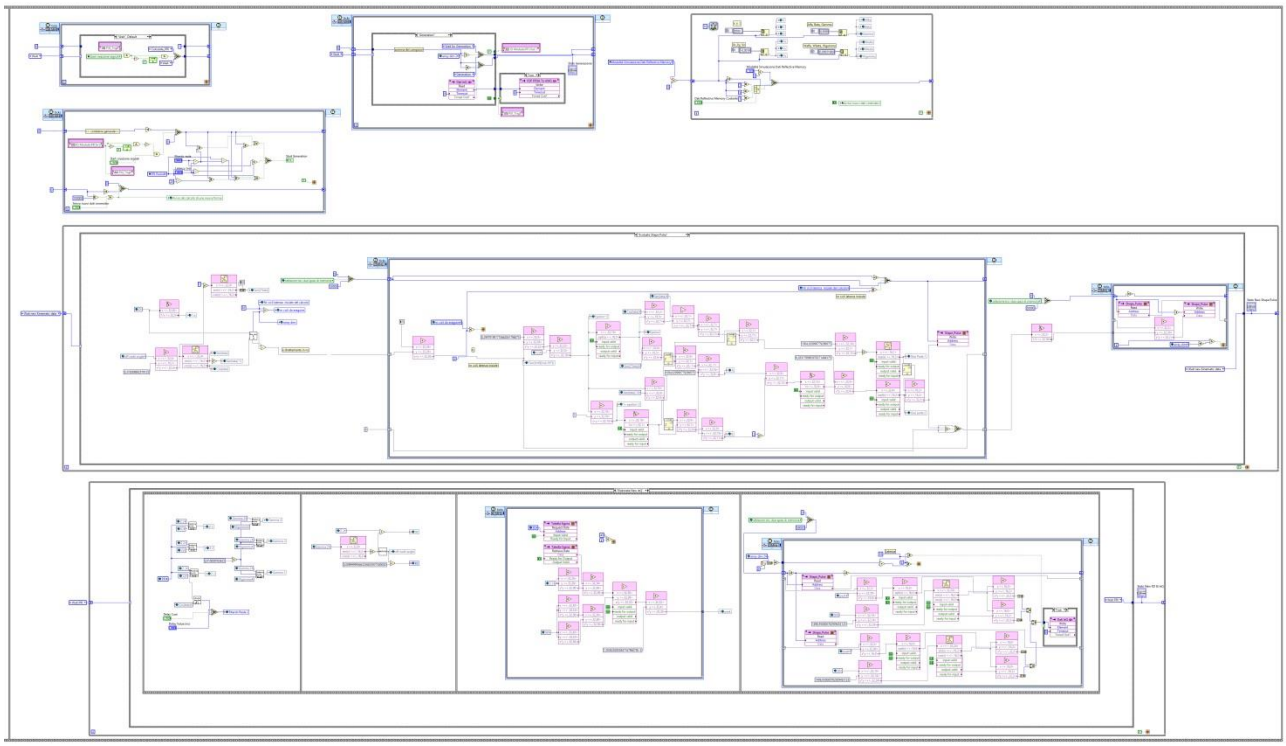

Figura 3.47 Codice, su FPGA, per l'elaborazione e l'invio dei campioni all'AWG

Il blocco successivo, in Figura 3.47, è il vero programma che, eseguito all'interno dell'FPGA, permette la temporizzazione e l'invio dell'eco elaborato all'AWG.
In questa sequenza, sono presenti quattro Single-Cycle Timed Loop e due cicli While. I sei cicli permettono di:

- valutare il periodo di PRI (in alto a sinistra)
- inviare i campioni all'AWG per la generazione (nel centro della prima riga)
- simulare l'aggiornamento dei dati cinematici, provenienti dal simulatore dell'interfaccia con il modulo EDM (in alto a destra)

- temporizzare la struttura generando un segnale, chiamato Generazione, per gestire il trasferimento dei campioni nell'AWG. (il secondo ciclo nella colonna di sinistra)
- calcolare i campioni dell'eco normalizzato (il timed loop al centro della figura)
- aggiornare i dati cinematici, calcolare la potenza reale e le componenti I&Q del segnale (il ciclo While in basso)

Questi cicli devono essere sempre eseguiti. Per questo, viene collegata una costante, con valore booleano falso, alla condizione di interruzione dei cicli. I cicli per il calcolare i campioni dell'eco normalizzato, per aggiornare i dati cinematici, calcolare la potenza reale e le componenti I&Q sono state presentate nei paragrafi precedenti. Queste elaborazioni sono utilizzate in due FSM distinte, composte da uno stato di attesa e uno di calcolo. Nello stato di Wait, il ciclo per il calcolo dell'eco normalizzato attende una variabile, controllata dall'unità di controllo, assuma un valore logico alto. Invece, l'altro ciclo attende un fronte di salita del PRI per avviare l'elaborazione dei dati cinematici e delle componenti I&Q. Una volta terminate le elaborazioni, le FSM ritornano nello stato di Wait.

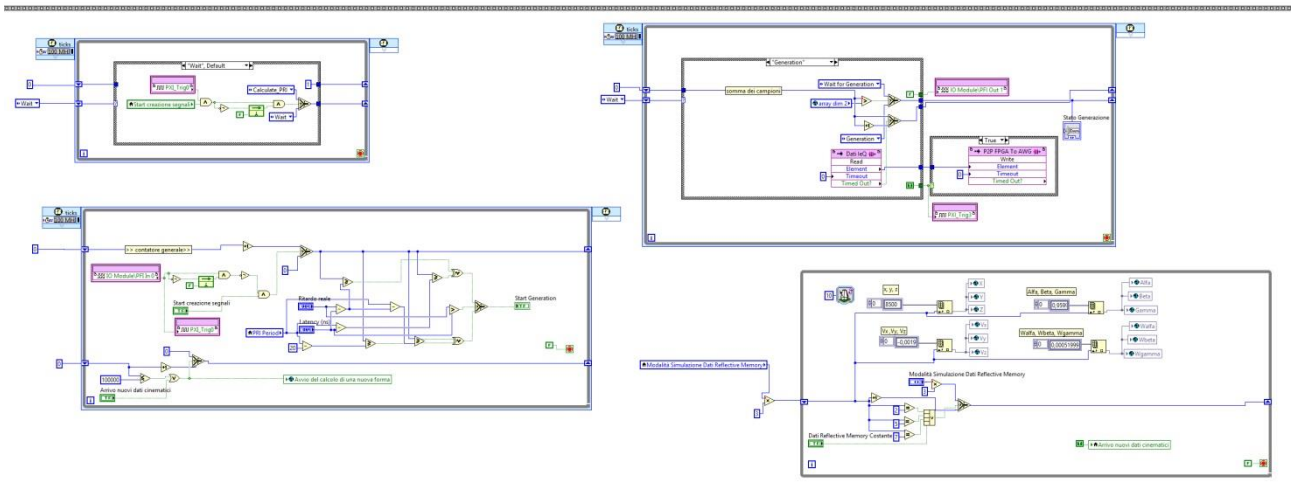

Figura 3.48 Codice, su FPGA, per la temporizzazione, in trasferimento dei dati e simulazione dell'aggiornamento dei dati cinematici

In Figura 3.48 vengono mostrati i restanti quattro cicli presenti nel VI FPGA. Il codice per valutare il periodo di PRI è identico a quello implementato nell'FPGA per l'acquisizione dei segnali. Questo è descritto come una FSM a tre stati. Nel primo si attende la prima transizione basso-alto del PRI, nel secondo si avvia il contatore e si controlla se viene ricevuto un nuovo fronte di salita sul PRI. Se viene ricevuto un nuovo fronte di salita si passa nel terzo stato, in cui viene letto il periodo di PRI e scritto in una variabile locale.

Il codice per inviare i campioni all'AWG è stato implementato con una FSM a tre stati. Lo stato inziale attende la prima transizione basso-alto del segnale Generazione. Una volta rilevata, la FSM può passare allo stato di generazione, il terzo stato, in cui avvia il trasferimento dei campioni dall'FPGA all'AWG. Una volta trasferiti tutti i campioni, la macchina a stati finiti passa al secondo stadio in cui si attende una nuova transizione basso-alto del segnale Generazione. Questo stato si differenzia dal primo per un comportamento diverso, verso la coda Peer To Peer. Nel primo stato, la coda Peer To Peer viene disabilitata, senza trasferire dati all'AWG. Nel secondo, invece, la coda Peer To Peer rimane attiva, ma s'inviano dei campioni nulli all'AWG. Quest'approccio è l'unico implementabile, per eseguire un controllo diretto da FPGA nella generazione dell'AWG.

Il blocco che si occupa della temporizzazione è stato implementato con l'utilizzo di alcuni Shift Register, di sommatori, di comparatori e di logica booleana. Il contatore generale utilizza uno Shift Register, inizializzato a zero. Questo contatore è incrementato se il segnale Start Creazione Segnali è abilitato, e se non viene rilevato un fronte di salita sul segnale di PRI. Nel caso contrario, il contatore viene portato a zero, mantenendo fermo il suo valore. Il PRI è collegato all'interfaccia digitale del NI 5781. Questo segnale viene anche inviato, attraverso una linea di trigger presente nel blackplane, all'altra FPGA per sincronizzare le esecuzioni. Utilizzando il contatore e conoscendo il periodo di clock, è possibile calcolare dei ritardi rispetto ad un fronte di salita del PRI, per portare ad un livello logico alto o basso i segnali di controllo del sistema.

In questo caso, si deve tener conto della latenza, per il trasferimento dei dati dall'FPGA all'AWG. Questa latenza, quantificata in 940ns, ci porta in due casi possibili. Il primo, in cui il valore del ritardo reale è maggiore del tempo di latenza. Allora, basta portare alto, il segnale per il trasferimento dei dati, con un ritardo fittizio pari alla differenza tra il ritardo reale e la latenza. Nell'altra condizione, in cui il ritardo sia minore della latenza, si considera un ritardo fittizio pari al periodo di PRI sottratto del ritardo reale e sommato della latenza. Ipotizzando che siano uguali due periodi di PRI consecutivi, si avvia il trasferimento dei dati all'AWG nel periodo di PRI precedente, evitando di imporre la limitazione di un ritardo minimo maggiore del periodo di clock. Nell'unità di controllo, viene generato il segnale Generazione che permette di avviare il trasferimento dei campioni all'AWG. Il primo comparatore controlla se il contatore sia maggiore del ritardo e, nel caso positivo, porta ad un livello logico alto il segnale. Il secondo comparatore e una porta Xor riportano ad un livello logico basso, il segnale Acquisizione, se viene superato un valore limite. Utilizzando un clock a 100MHz, il ritardo dal fronte di salita del PRI può essere solo un multiplo del periodo di clock, ossia 10ns.

L'ultima porzione di codice da presentare è quello destinato alla simulazione dell'aggiornamento dei dati cinematici, provenienti dal simulatore dell'interfaccia con il modulo EDM. Questo blocco è stato implementato con un ciclo While non temporizzato. Per dare una temporizzazione al ciclo, è stata inserita la funzione Loop Timers, che permette di definire il tempo di esecuzione minimo per ogni iterazione del ciclo. Al ogni iterazione, quindi, si aggiorneranno i dati cinematici e una variabile booleana, che segnala la ricezione di nuovi dati cinematici.

Attraverso questo VI FPGA e il codice descritto al suo interno, è stato possibile implementare il calcolo e la generazione di un eco simulato.

Programma eseguito sul processore

Il VI, eseguito sul processore, fornisce l'interfaccia tra i VI FPGA e l'utente. Inoltre permette, in maniera semplice, il salvataggio dei dati acquisiti. Lo schema a blocchi, mostrato in Figura 3.49, evidenzia le varie fasi che compongo il programma.

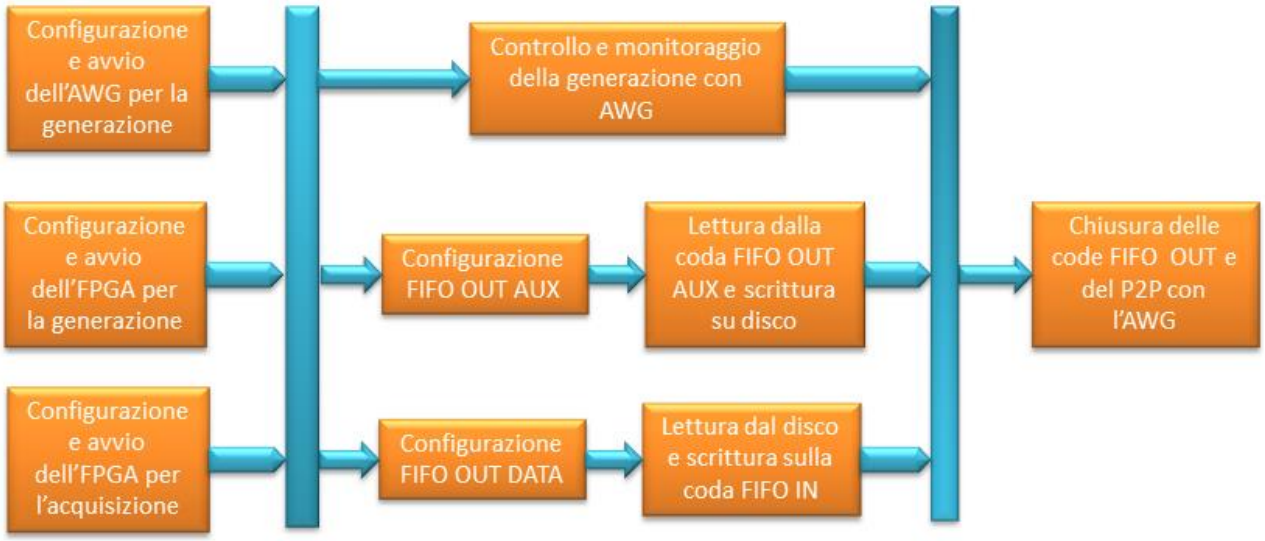

Figura 3.49 Schema a blocchi del programma eseguito sul sistema Host

Nella fase iniziale del VI, viene avviata la comunicazione con i VI FPGA e configurato l'AWG. In seguito, vengono configurate le code FIFO OUT, necessarie per il trasferimento dei dati acquisiti. Completate le operazioni di configurazione, è possibile avviare tre blocchi di codice che saranno eseguiti iterativamente. Nel primo, viene eseguito un controllo e un monitoraggio dell'FPGA di generazione e dell'AWG. Il secondo, permette di controllare e monitorare l'FPGA destinata alla generazione. Inoltre, eseguire un polling sulla FIFO OUT AUX, controllando se sono presenti nuovi dati. Se la coda FIFO non è vuota, si avvia la scrittura dei dati ausiliari su disco. L'ultimo ciclo eseguirà sempre il polling, ma sull'altra FIFO, destinata al trasferimento dei campioni acquisiti. Se sono presenti nuovi dati, esegue la scrittura dei campioni su disco. La formattazione dei dati avviene direttamente su FPGA. Quest'approccio non complica o appesantisce il VI FPGA, ma alleggerisce quello sul processore. In questo modo, si evita che il processo di formattazione dei dati, se fosse eseguito sul processore, sia messo in attesa dal SO, impegnato in altre operazioni.

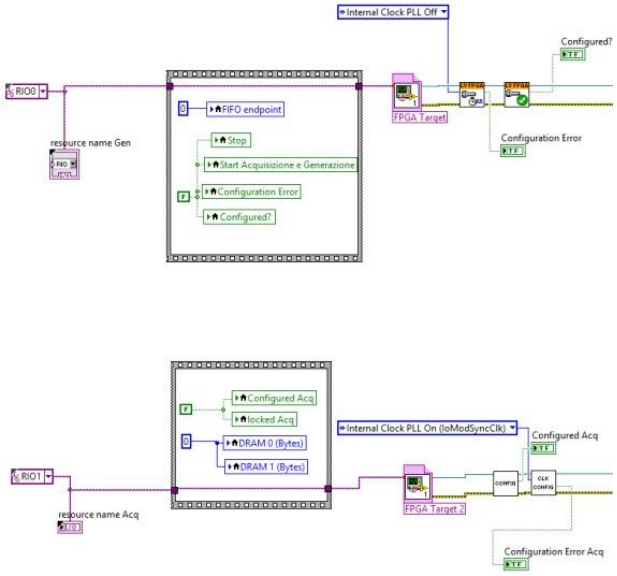

Figura 3.50 Codice per avviare la comunicazione tra il VI Host e i VI FPGA

Nella Figura 3.50, è mostrato il codice per avviare la comunicazione con i due VI FPGA. I blocchi successivi permettono una corretta configurazione dei moduli adapter NI5761 e NI5781. Completata la configurazione e se non ci sono stati errori, può essere eseguito il codice mostrato in Figura 3.51.

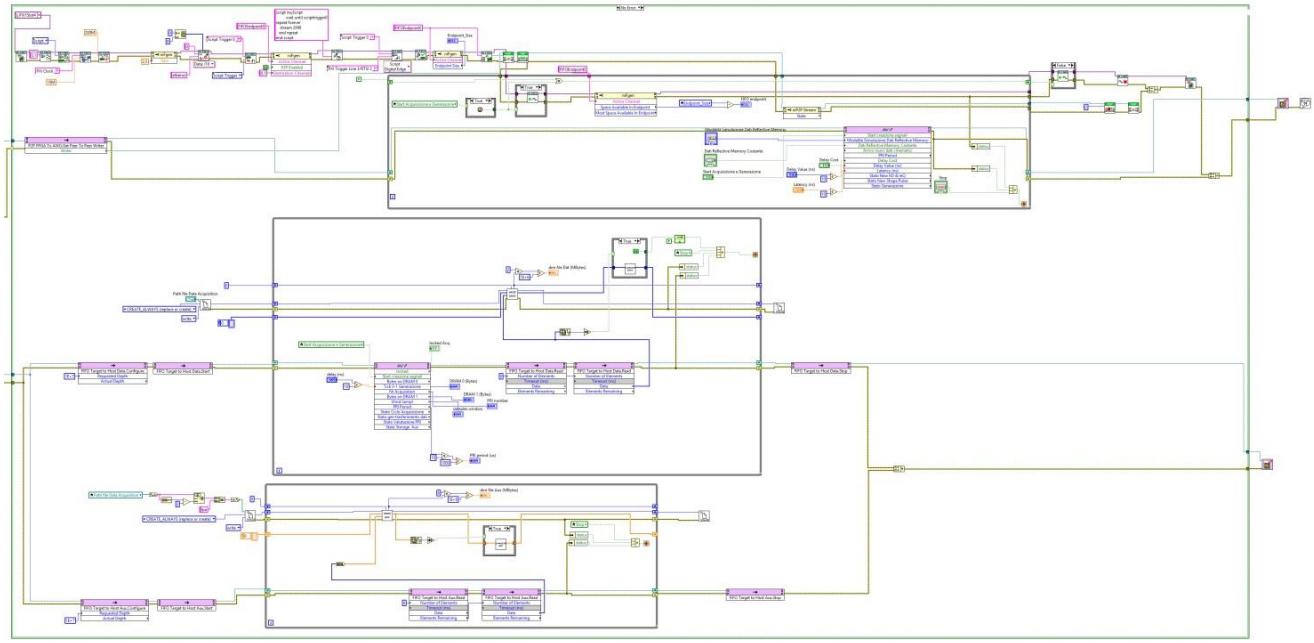

Figura 3.51 Codice, eseguito sul processore, per il controllo delle FPGA e la scrittura su disco dei campioni

La prima operazione eseguita è la configurazione e l'avvio dell'AWG. Il codice necessario per la configurazione è mostrato, in maggiore dettaglio, nella Figura 3.52.

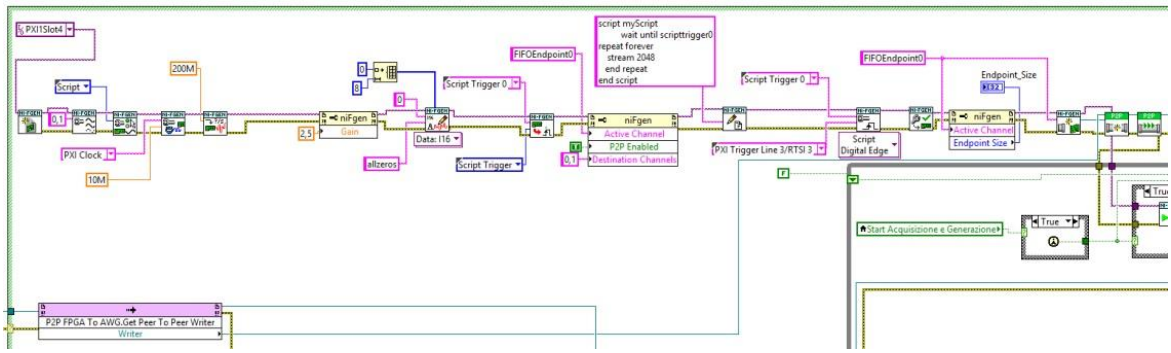

Figura 3.52 Codice, su Host, per la configurazione dell'AWG

Le operazioni eseguite, nell'ordine in cui sono rappresentate in Figura 3.52, riguardano:

- l'apertura della comunicazione con il modulo PXIe NI 5451
- la configurazione dei canali d'uscita, nel nostro caso i canali 0 e 1
- la scelta della modalità di esecuzione (per utilizzare una comunicazione Peer To Peer, si deve selezionare la modalità script)
- la configurazione del clock a 200MHz sincrono con il PXI Clock a 10MHz

- la definizione dello script da eseguire sull'AWG (viene utilizzato il linguaggio script per programmare l'AWG)
- la scelta del trigger per l'avvio della generazione
- la creazione e l'avvio del collegamento Peer To Peer tra l'FPGA e l'AWG (dall'FPGA viene prelevato il riferimento Peer To Peer Writer e dall'AWG il riferimento Peer To Peer Reader)

Nella Figura 3.51, si possono notare i tre cicli While, in cui sono eseguite le operazioni sul processore. Il primo in alto permette di leggere e scrivere dei controlli e degli indicatori del Front Panel del VI FPGA. In questo modo, si controlla l'FPGA destinata alla generazione. Quello al centro permette, in maniera analoga al primo ciclo, il controllo dell'FPGA di acquisizione. Inoltre permette di controllare se sono presenti nuovi dati ausiliari e, in caso affermativo, la loro scrittura su disco. Quello in basso, esegue il polling sulla coda FIFO OUT DATA per consentire il salvataggio dei dati su disco. Per la scrittura vera e propria, sono utilizzati dei SubVI, che saranno descritti nel paragrafo successivo. Il programma completa la sua esecuzione se occorre un problema nell'AWG, o nel collegamento con l'FPGA o nella scrittura dei file. Inoltre, il VI può essere terminato dall'utente, premendo il pulsante STOP, presente nel Front Panel del VI Host. Prima che il VI termina la propria esecuzione, vengono chiusi i riferimenti ai VI FPGA, la sessione di generazione dell'AWG e le code FIFO DMA utilizzate per il trasferimento dei dati.

Scrittura dei file su disco

Il SubVI, eseguito nel programma Host, effettua la scrittura dei dati su disco. Le operazioni di scrittura sono effettuate senza l'apertura e la chiusura del file, perché viene utilizzato un handle (un riferimento) generato in precedenza. Evitando di aprire e chiudere il file in ogni iterazione, si velocizza la scrittura dei dati su disco.

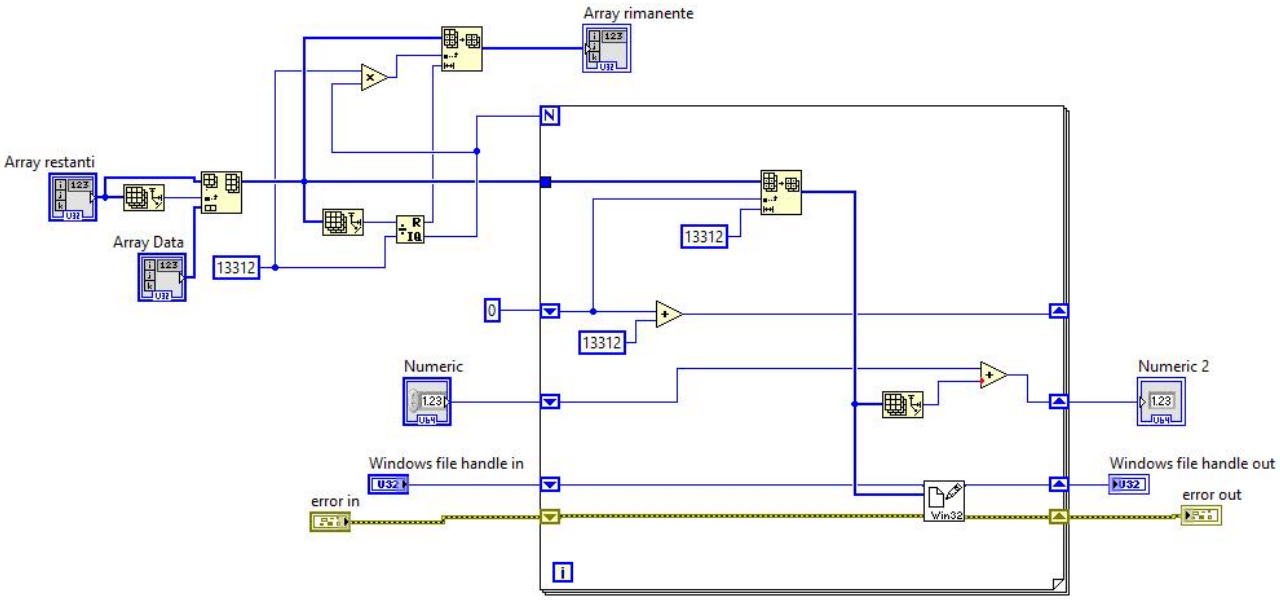

Figura 3.53 Block Diagram per la scrittura dei campioni acquisiti

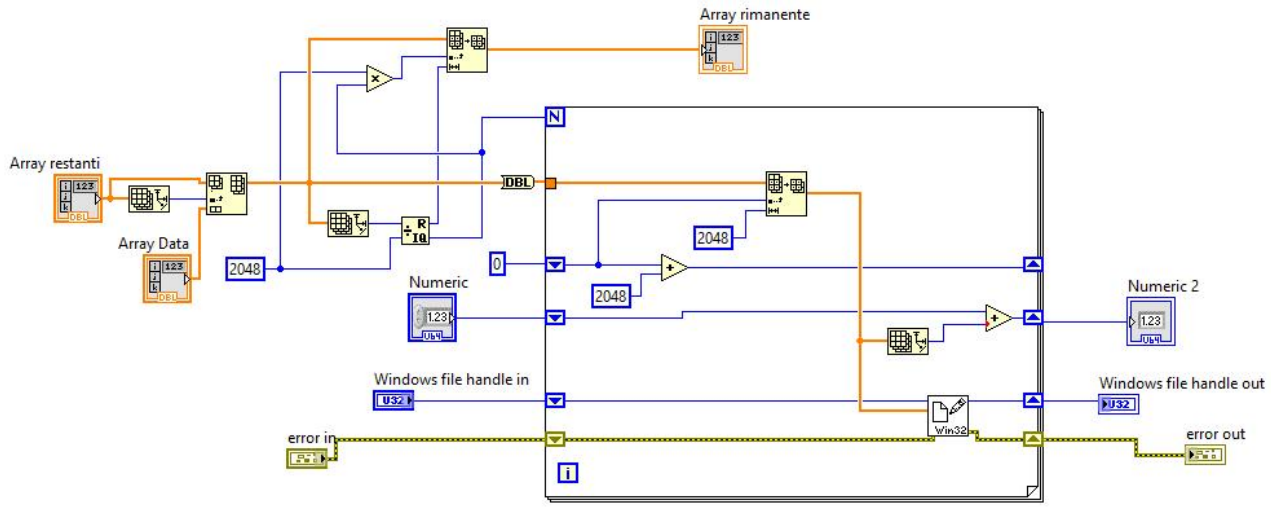

Figura 3.54 Block Diagram per la scrittura dei dati ausiliari

I due SubVI, mostrati nella Figura 3.53 e Figura 3.54, permettono la memorizzazione dei dati letti dalla coda. Le operazioni eseguite sui due programmi sono identiche. In ognuno, viene eseguita la scrittura su disco, di un blocco di dimensione fissa di dati. Per non appesantire le operazioni di scrittura, vengono utilizzate le dimensioni di 13312, per i campioni acquisiti, e di 2048, per i dati ausiliari. Leggendo un numero casuale di campioni dalla FIFO, è probabile che non sia letto un numero di campioni multiplo alla dimensione fissata. Per questo, sono presenti due array in ingresso: uno per i dati provenienti dalla FIFO, nel ciclo corrente, e un altro per i dati, da accessi precedenti alla FIFO e non ancora salvati. Il VI esegue l'unione dei dati concatenandoli. I campioni rimanenti sono messi all'inizio dell'array, perché provengono da acquisizioni precedenti, rispetto a quelli appena letti dalla FIFO.

Altri parametri d'ingresso dei SubVI sono il numero di acquisizioni scritte, per implementare un contatore di acquisizioni, e l'Handle di Windows, per la gestione del file.

Una volta concatenati i dati, viene eseguito un ciclo for per la scrittura sul disco. Ad ogni iterazione del ciclo for, corrisponde la scrittura su disco di un intero blocco di dati. Il ciclo viene eseguito un numero variabile di volte, poiché non viene letto sempre lo stesso numero di campioni. Il rapporto tra il numero di campioni da processare e il numero di campioni per ogni blocco rappresenta il numero d'iterazioni del ciclo for da eseguire.

Le funzioni per la gestione dei file

Le funzioni, utilizzate all'interno dei VI per la gestione dei file, mostrate in Figura 3.55, sono delle particolari funzioni avanzate disponibili sul sito National Instruments. Queste funzioni utilizzano le chiamate di sistema Win32, permettendo prestazioni migliori, in termine di velocità di trasferimento, rispetto alle funzioni di base di Labview.

Possono essere utilizzate come SubVI, ma possono essere eseguite solo sul processore perché utilizzano le chiamate di sistema, disponibili solo in un Sistema Operativo.

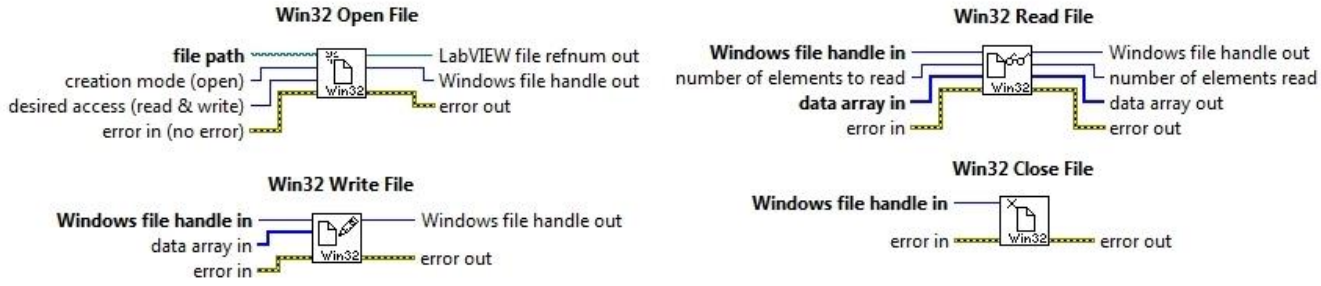

Figura 3.55 Funzioni di gestione dei file attraverso le chiamate di sistema Win32

Le misure delle prestazioni con le funzioni Win32 e con le funzioni standard Labview, per la gestione dei file, sono state eseguite con un programma di test. Nei test, viene rilevata la velocità di trasferimento su disco, scrivendo 41943040 elementi. La quantità di elementi scritti varia in base al tipo di dati utilizzato. Se si utilizza una rappresentazione di interi a 8bit vengono scritti 40MB contro gli 80MB scritti nel caso di una rappresentazione di interi a 16 bit. Questa differenza è dovuta alla quantità d'informazioni salvate per ogni dato. Le prestazioni, rilevate con prove sperimentali, vengono mostrate in Figura 3.56.

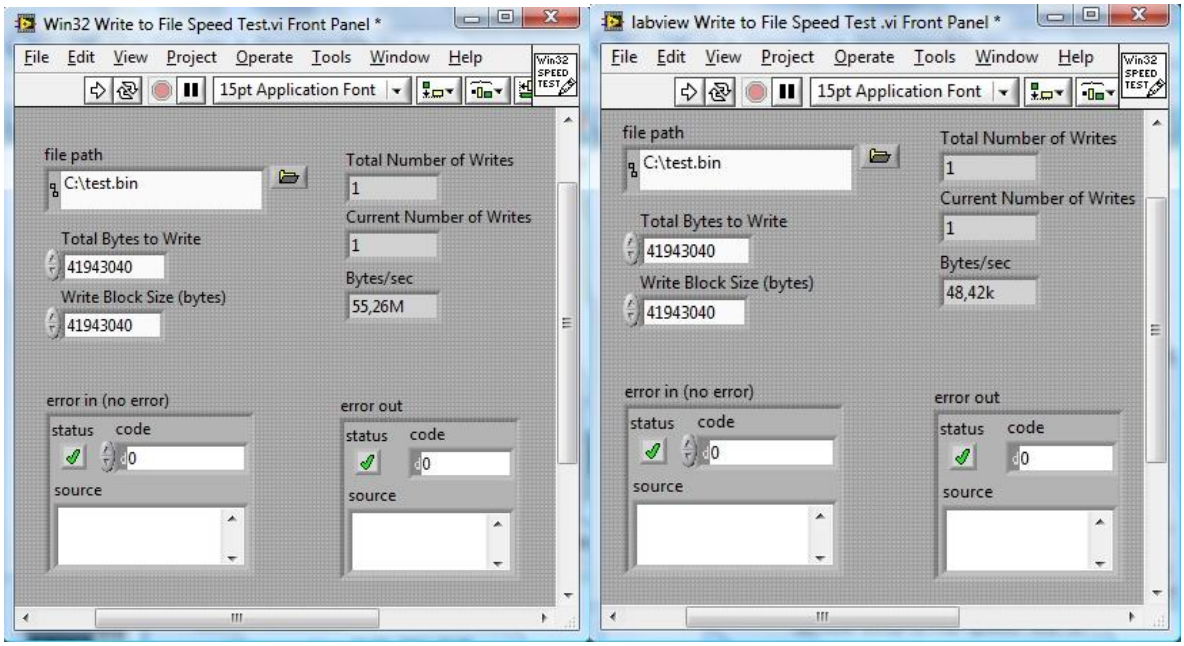

Figura 3.56 Front panel del programma per i test della velocità di scrittura. A sinistra per le funzioni Win32 e a destra per le funzioni Labview

Logica utilizzata Su Fpga per L'implementazione

Nel sistema Host, che ospita un sistema operativo Windows 7, la compilazione dei programmi, appena descritti, impiegano all'incirca 45 minuti, per quello di acquisizione e circa 90 minuti, per il programma di generazione. Per la compilazione, la parte più lunga riguarda il compilatore Xilinx, che genera, dai file intermedi di Labview FPGA, il file Bitstream per la programmazione dell'FPGA.

Non è possibile né velocizzare, né personalizzare quest'operazione. È possibile scegliere, come unica opzione in fase di compilazione, l'implementazione di parti di codice che si ripetono nel VI. Per effettuare tale scelta, si seleziona la voce Reentrant execution, nella finestra delle proprietà del VI FPGA, nella categoria Execution.

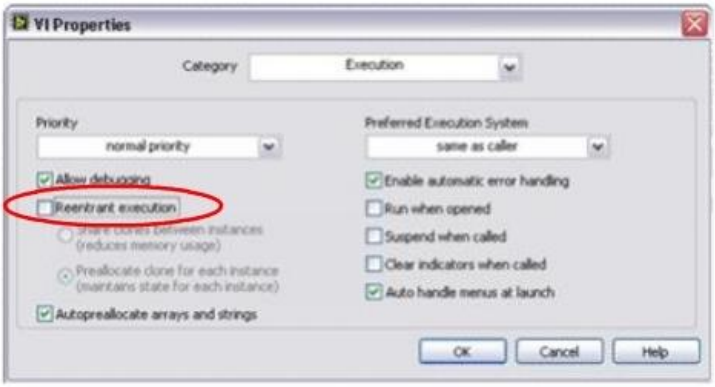

Figura 3.57 Finestra delle proprietà del VI FPGA e opzione Reentrant execution non selezionata

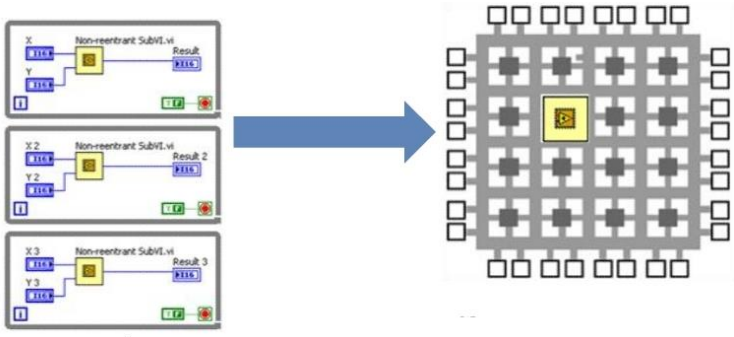

Figura 3.58 Corrispondente circuitale del codice nel caso della Figura 3.57

Se questa opzione non è attiva, viene implementata solo una copia del codice su FPGA. Se sono presenti più istanze dello stesso SubVI, le risorse Hardware utilizzate, diventano degli elementi condivisi tra le varie istanze. Questa soluzione impedisce l'esecuzione parallela di più istanze del SubVI, per l'assenza di risorse hardware. Il vantaggio si può individuare nella diminuzione della logica utilizzata sul dispositivo.

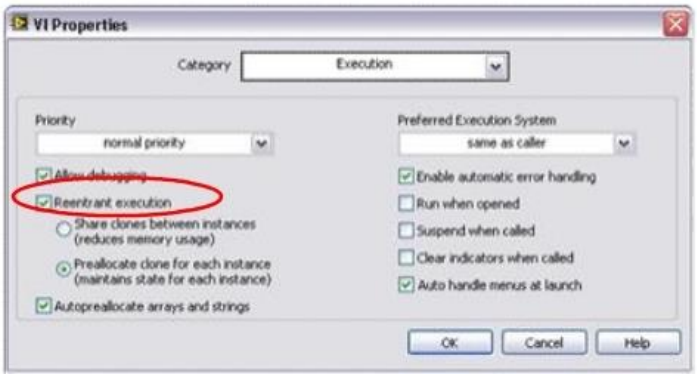

Figura 3.59 Finestra delle proprietà del VI FPGA e opzione Reentrant execution attiva

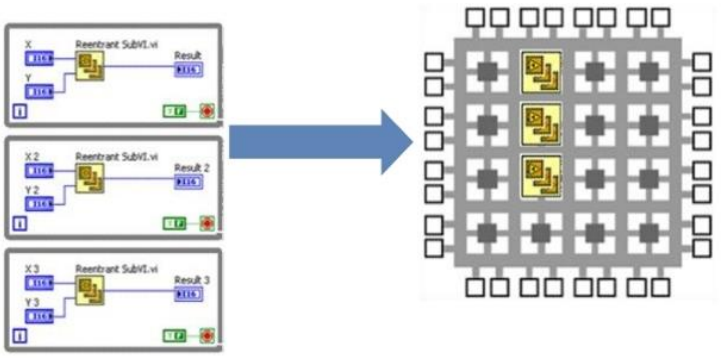

Figura 3.60 Corrispondente circuitale del codice nel caso della Figura 3.59

Se viene scelta la Reentrant Execution, come in Figura 3.59, vengono implementate più copie dello stesso codice su FPGA. In questo caso, ogni istanza ha a disposizione le risorse hardware per l'esecuzione in parallelo dei SubVI, migliorando la velocità d'esecuzione. Nei VI FPGA sviluppati, viene utilizzata la tecnica del codice rientrante. Così ogni singola istanza di un SubVI verrà allocata autonomamente.

Il compilatore visualizza, dopo aver effettuato tutte le operazioni di compilazione e la creazione del file Bitstream, le risorse logiche e di memoria utilizzate su FPGA per implementare i VI.

All'interno del report di compilazione, chiamato XilinxLog, si possono visualizzare maggiori dettagli sulla logica utilizzata sotto la voce "Design Summary" generato dopo il processo di Map. Qui viene indicato quante SliceM, ossia le slice che presentano dei registri al loro interno, e quante SliceL, slice mirate all'utilizzo della logica combinatoria, vengono utilizzate. Inoltre sono specificate le modalità di utilizzo delle varie slice. Le SliceM sono utilizzate normalmente, come Flip Flop. Le Slice L, invece, come logica, come memoria e come Route-Thru esclusivo. Le slice, impiegate come Route-Thru, rappresentano quelle utilizzate per eseguire l'accesso a punti interni alle slice, quando un accesso diretto non è disponibile o è meno efficiente. Per le SliceL utilizzate come memoria, viene specificato se sono utilizzate come RAM a due porte o come Shift Register.

Altre risorse, presenti nell'FPGA, vengono utilizzate per routing del clock, per i blocchi di IO dell'FPGA e di startup del dispositivo. Per le risorse di routing del segnale di clock sono presenti:

- 32 BUFG, un'architettura indipendente per il buffer globale e per avere un elevato fan-out per i segnali di clock
- 10 BUFGCTRL, un buffer del clock globale, designato come un multiplexer 2:1 sincrono/asincrono senza glitch
- 32 BUFIO, un buffer che permette il timing senza l'uso di DCM, Digital Clock Managers
- 80 BUFR, che permettono di utilizzare un clock, ad alto fan-out, in una regione del clock indipendente dall'albero di clock generale
- 12 DCM_ADV, dei Digital Clock Managers per effettuare il phase shifting e per una riconfigurazione dinamica

Altre risorse sono destinate alle interfacce di IO, presenti sull'FPGA. Per ogni terminale di IO, è presente una colonna in cui sono contenuti due IOB, due ILOGIC, due OLOGIC e due IODELAY.
È presente nello XilinxLog, una parte dettagliata, indicata come "Advanced HDL Synthesis Report", in cui sono elencati i dispositivi utilizzati come addizionatori, contatori, accumulatori,

registri, comparatori, multiplexer, decoder e l'operazione booleana Xor. Queste macro statistiche sono redatte dal compilatore dopo la fase di sintesi avanzata del codice HDL, ma prima della fase di mapping delle risorse sull'FPGA. Dopo la fase di Place and Route del compilatore, viene generato un report di timing. In questo report, viene descritta la frequenza massima cui può lavorare il codice.

Sull'FPGA per la generazione vengono utilizzati:

- Slice Totali: 69,6% (10246 di 14720)
- Slice con Registri (SLICEM): 43,2% (25448 di 58880)
    - Usati come Flip Flop: 25448
- Slice con LUT (SLICEL): 42,3% (24897 di 58880)
    - Usati come logica: 24265 di 58880
    - Usati come Memoria: 481 di 24320
        - Come RAM Dual Port: 132
        - Come Shift Register: 349
    - Usati come Route-Thru esclusivo: 151
- DSP48: 20,8% (133 di 640)
- Blocchi di RAM: 70,9% (173 di 244)

In questo caso sono stati utilizzati, dal report avanzato della sintesi HDL dallo XilinxLog:

- RAM : 17
- Moltiplicatori : 16
- Addizionatori/Sottrattori : 704
- Contatori : 31
- Accumulatori : 4
- Registri come Flip Flop : 33648
- Comparatori : 938
- Multiplexer : 4604
- Decoder : 2
- Macchine a Stati finiti : 50
- Xor : 437
- Registri per le FSM : 20869
- Shift Register : 317

Viene riportato, infine, il Report riguardante il timing:

- 40 MHz Onboard Clock: 40,00 MHz (42,68 MHz massimo)
- 100 MHz Clock: 100,00 MHz (106,93 MHz massimo)
- 25MHz: 25,00 MHz (27,14 MHz massimo)
- TS_BusClk: 128,06 MHz (massimo)
- TS_SlowBusClk: 148,13 MHz (massimo)
- TS_DramClk200: 205,13 MHz (massimo)
- TS_DramClk200s90: 212,18 MHz (massimo)

- TS_IoRxClock: 333,33 MHz (massimo)
- TS_ClockGenXilinxV5x_TxDcm_TxHighSpeedClkDcm: 300,12 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxHighSpeedClkDcm: 450,25 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxLowSpeedClkDcm: 159,29 MHz (massimo)

Sull'FPGA per l'acquisizione vengono utilizzati:

- Slice Totali: 46,0% (6767 di 14720)
- Slice con Registri (SLICEM): 22,1% (13030 di 58880)
    - Usati come Flip Flop: 13030
- Slice con LUT (SLICEL): 24,0% (14130 di 58880)
    - Usati come logica: 13862 di 58880
    - Usati come Memoria: 204 di 24320
        - Come RAM Dual Port: 132
        - Come Shift Register: 72
    - Usati come Route-Thru esclusivo: 64
- DSP48: 0,0% (0 di 640)
- Blocchi di RAM: 44,3% (108 di 244)

In questo caso sono stati utilizzati, dal report avanzato della sintesi HDL dallo XilinxLog:

- RAM : 20
- Moltiplicatori : 0
- Addizionatori/Sottrattori : 59
- Contatori : 35
- Accumulatori : 4
- Registri come Flip Flop : 8097
- Comparatori : 39
- Multiplexer : 1304
- Decoder : 4
- Macchine a Stati finiti : 27
- Xor : 133
- Registri per le FSM : 6896
- Shift Register : 44

Viene riportato, infine, il Report riguardante il timing:

- 40 MHz Onboard Clock: 40,00 MHz (86,19 MHz massimo)
- 100 MHz Clock: 100,00 MHz (100,64 MHz massimo)
- TS_AdcDataClk: 428,45 MHz (massimo)
- TS_BusClk: 125,53 MHz (massimo)
- TS_SlowBusClk: 136,26 MHz (massimo)
- TS_DramClk200: 205,59 MHz (massimo)
- TS_DramClk200s90: 223,76 MHz (massimo)

- TS_IoRxClock: 333,33 MHz (massimo)
- TS_Puma20Window_theCLIPs_IO_Module_CLIP2_NI5761Topx_Ni5761AdcSamplerx_Ni5761TimingEnginex_SampleClkPb: 256,48 MHz (massimo)
- TS_Puma20Window_theCLIPs_IO_Module_CLIP2_NI5761Topx_Ni5761AdcSamplerx_Ni5761TimingEnginex_DivSampleClkPb: 156,35 MHz (massimo)
- TS_ClockGenXilinxV5x_TxDcm_TxHighSpeedClkDcm: 286,53 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxHighSpeedClkDcm: 450,25 MHz (massimo)
- TS_ClockGenXilinxV5x_RxDcm_RxLowSpeedClkDcm: 159,97 MHz (massimo)

Dai report si evince che, nell'FPGA utilizzata per la generazione, sono utilizzate più risorse. Inoltre, è interessante rilevare che il valore massimo relativo al clock a 25MHz, utilizzato per il calcolo dell'eco normalizzato, diminuisce. Questo perché, nel caso in cui è stato compilato solo il codice per l'elaborazione dei campioni, non viene implementato altro codice, occupando le risorse dell'FPGA. L'implementazione del codice, in una FPGA più occupata, è più difficoltosa e meno veloce. Un altro aspetto importante riguarda l'utilizzo dei blocchi di RAM. Questi sono utilizzati maggiormente nell'FPGA per la generazione perché non viene sfruttata la DRAM, utilizzata in quello per l'acquisizione.

4. Analisi Sperimentali del Sistema di Test

Utilizzo del sistema

Il programma è controllato direttamente dal Front Panel del VI Host, eseguito sul processore, che risulta interfaccia utente. Da questo VI, sono richiamati tutti gli altri, per una corretta esecuzione. In Figura 4.1, è mostrata un'immagine dell'interfaccia utente.

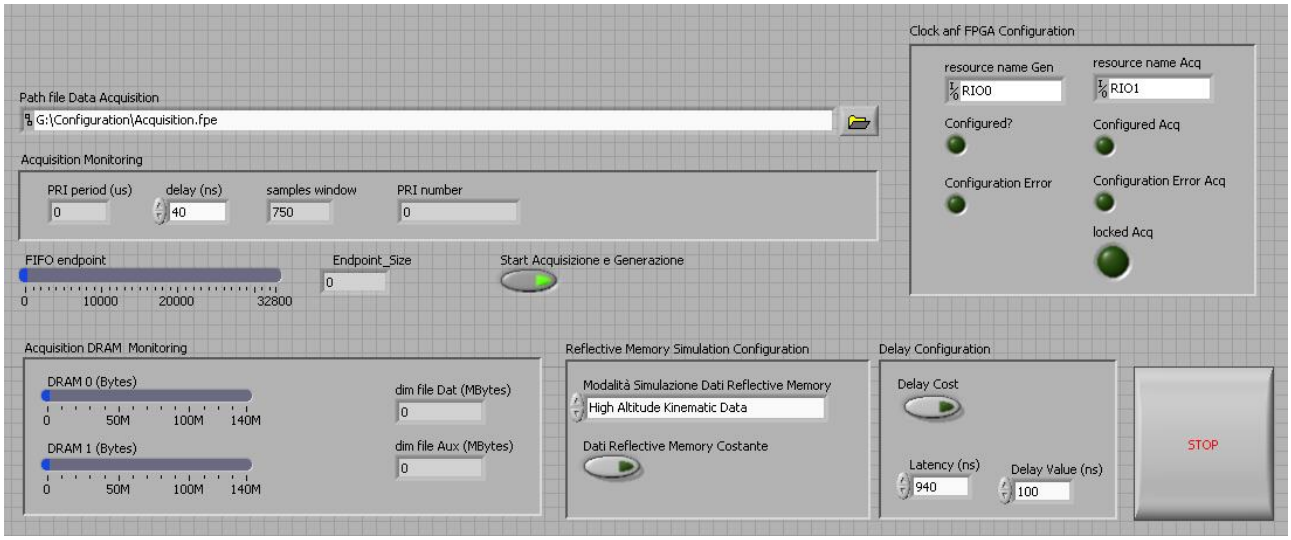

Figura 4.1 Front Panel del VI Host, interfaccia dell'EG

Premendo Run dal menu a tendina Operate, o dal tasto evidenziato con il contorno di una freccia vuota, oppure con i tasti di scelta rapida Ctrl+R, è possibile avviare l'esecuzione del programma.
Prima di questo passo, si devono configurare alcuni parametri, per un corretto funzionamento. I parametri, necessari per configurare una corretta esecuzione, sono:

- Nel pannello clock and FPGA Configuration, i terminali "resource name", con cui scegliere quale FPGA utilizzare per le operazioni di generazione e acquisizione.
- Nel pannello Acquisition monitoring, il controllo "delay (ns)" permette di introdurre un ritardo, tra la ricezione del segnale di PRI e l'avvio dell'acquisizione. Nella configurazione di default, come mostrato in Figura 4.1, è inserito un ritardo di 40ns.
- Nel pannello Delay monitoring, i controlli "Delay Cost" e "Delay Value (ns)" permettono di selezionare un ritardo costante, indipendente dai dati cinematici del radar, tra la ricezione del segnale di PRI e l'avvio della generazione.
- Il controllo "Start Acquisizione e Generazione" permette di abilitare, o disabilitare, la lettura del segnale di PRI (per esempio, inviato per errore all'EG).
- Il controllo "Path File Data Acquisition", in cui è possibile specificare la locazione di memoria e il nome del file, per il salvataggio dei campioni acquisiti.
- Il tasto STOP, che permette di interrompere correttamente l'esecuzione.

- Nel pannello Reflective Memory Simulation Configuration, la modalità di simulazione dei dati provenienti dal simulatore dell'interfaccia con il modulo EDM.

Per la simulazione dei dati provenienti dalla reflective memory, sono state implementate tre possibili scenari. Ognuno, può essere utilizzato con un opportuno PRI, da fornire al sistema dall'NI 5781. Si deve selezionare l'opportuno scenario, tra High Altitude Kinematic Data, Medium Altitude Kinematic Data e Low Altitude Kinematic Data, per valutare un'eco coerente con il PRI e con la posizione del radar. Nel caso in cui non viene selezionato il valore corretto, è possibile che la durata dell'impulso calcolato, generato a 200MHz, sia maggiore del periodo di PRI. Questa condizione potrebbe portare alla perdita d'impulsi del PRI e ad un errore nelle operazioni di generazioni. Anche in caso di errore, il processo di acquisizione, poiché implementato separatamente, potrà essere eseguito regolarmente. Una volta completata la configurazione, è possibile premere il tasto Run, per iniziare l'esecuzione del programma. Per abilitare la lettura del PRI, il controllo Start Creazione Segnali deve assumere un valore positivo. Solo in questo modo, è possibile avviare l'acquisizione, la scrittura su disco e la generazione.

L'indicatore PRI Number, che indica il numero di PRI ricevuti dopo che Start Creazione Segnali è stato portato alto, permette di capire quando si esegue un'acquisizione. Se l'indicatore mostra un valore crescente, si sta acquisendo il segnale. Per l'operazione di generazione, possiamo controllare l'indicatore FIFO Endpoint. Se l'indicatore si muove, vuol dire che si trasferiscono i dati all'AWG, quindi l'FPGA di generazione funziona perfettamente.

Gli altri indicatori presenti nel front panel sono:

- Configured?, Configuration Error, Configured Acq,Configuration Error Acq e locked Acq, che permettono di capire se i moduli sono stati inizializzati correttamente, o se si è verificato un errore
- PRI period (us) visualizza il valore di PRI rilevato dallFPGA per l'acquisizione
- Sample window visualizza il numero di campioni acquisito, per ogni periodo di PRI
- PRI number rappresenta il numero di acquisizioni eseguite su FPGA
- FIFO endpoint rappresenta lo spazio occupato sulla memoria RAM (512MB), presente nell'AWG.
- Nel pannello Acquisition DRAM Monitoring, le slice DRAM 0 (Bytes) e DRAM 1 (Bytes) indicano lo spazio occupato sui due banchi di DRAM, presenti nell'FPGA d'acquisizione
- Dim file Dat (MB) e Dim file Aux (MB) indicano la dimensione dei file in scrittura.

Dispositivi utilizzati per le simulazioni

Sono stati utilizzati, per la simulazione e il test dell'EG, degli strumenti di misura e alcuni generatori. Sono stati utilizzati:

- Un oscilloscopio Tektronix TDS784, per la visualizzazione dei segnali generati e dei segnali di riferimento
- Un generatore Tektronix AWG2041, per la simulazione del segnale di PRI
- Un Generatore HP 33120A, per la generazione di forme d'onda per l'acquisizione

Oscilloscopio Tektronix TDS784

Il Tektronix TDS784 è un oscilloscopio per l'acquisizione di segnali analogici. Ha la possibilità di acquisire quattro segnali analogici, contemporaneamente, a 4GS/s.

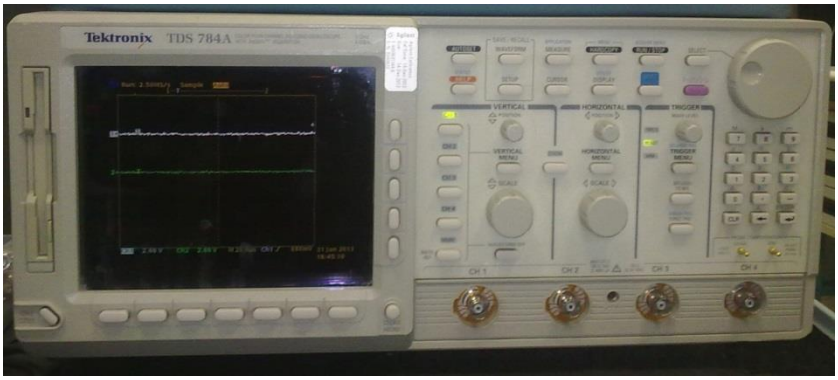

Figura 4.2 Oscilloscopio Tektronix TDS784

Questo strumento ha una banda di 1GHz, una sensibilità verticale da 1mV/div fino a 10V/div. Ha una risoluzione di 8bit, migliorabile fino a 13bit in alta risoluzione. La memoria interna del TDS784 permette di memorizzare 500000 per la visualizzazione e il trasferimento dei dati su di una postazione remota. È presente, nella parte sinistra del pannello frontale, un lettore di Floppy Disk da 3,5 pollici, con cui è possibile salvare, ciò che viene visualizzato sul display. Questa unità semplifica il salvataggio dei dati, evitando di dover interfacciare l'oscilloscopio con una stazione remota, tipo un pc.

Tektronix AWG2041

L'AWG 2041 è un generatore di forme d'onda arbitrarie, capace di generare sia forme d'onda standard che forme d'onda arbitrarie. Ha un canale d'uscita, CH1, con una risoluzione di 8bit e un ulteriore canale per la generazione del segnale negato dispetto a quello su CH1.

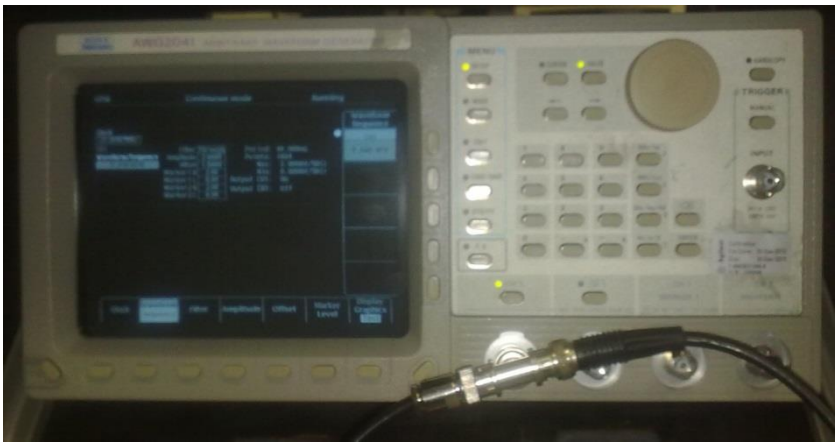

Figura 4.3 Generatore di forme d'onda arbitrarie AWG2041

Le forme d'onda possono essere

- create dall'editor grafico, presente nel dispositivo,
- generate dalle equazioni, create con l'editor di equazioni,

- trasferite dalle interfacce RS-232 o GPIB,
- trasferito da un oscilloscopio o da un altro generatore di forme d’onda arbitrarie

Il dispositivo permette la generazione continua di forme d’onda da vari file, specificati in una sequenza. La frequenza di generazione dei campioni può essere selezionata tra un clock interno tra 1KHz e 1,024GHz, o un clock esterno. Il generatore ha una memoria interna di 1MB, in cui registrare le forme d’onda per la generazione; 4MB, in cui sono memorizzate le forme d’onda standard e una memoria di 512KB, in cui memorizzare le forme d’onda arbitrarie.

Generatore HP 33120A

L'HP 33120A è un sintetizzatore di funzioni, con la possibilità di generare forme d’onda standard, incorporate nel generatore, e di forme d'onda arbitrarie.

Figura 4.4 Generatore HP33120A

Lo strumento è in grado di generare 10 forme d'onda standard ed ha 4 memorie, per forme d'onda arbitrarie (a 12bit e 40 MSa/s). Tutte le funzioni, sia standard che arbitrarie, sono modulabili in ampiezza e in frequenza. L'impedenza di uscita dello strumento è di 50Ω ed è possibile specificare la modalità di lavoro dell'uscita, cioè in adattamento di carico (resistenza di carico di 50Ω) o su circuito aperto. Questo strumento è controllabile tramite l'interfaccia HPIB da un computer, in modo da costruire, su quest'ultimo, la forma d'onda desiderata e in seguito, trasferirla al generatore. Nei test effettuati, l'HP 33120A è stato utilizzato per generare i segnali da acquisire.

Visualizzazione dei dati scritti su disco

A ogni impulso di PRI, è realizzato il campionamento e la scrittura su disco di un numero variabile di campioni. I dati sono inviati al processore, unica unità capace di scrivere i dati su disco, dall’FPGA, che svolge il ruolo d’intermediario tra il segnale acquisito, digitalizzato dal modulo NI 5761, e il processore. Il numero di campioni acquisiti, dipendenti dalla quota del radar, sono mostrati nella Figura 4.5. Nelle acquisizioni è stata utilizzata una frequenza di campionamento di 250MHz, la massima utilizzabile dall’NI 5761. In questo modo, si vuole portare NI 5761 alla frequenza massima di funzionamento.

Periodo di PRI inviato dal radar [µs]	Numero di campioni da acquisire tra due PRI	Tempo di acquisizione effettiva con f_{ck} di 200MHz [µs]	Percentuale del tempo d'acquisizione, rispetto al periodo di PRI	Velocità di trasferimento dall'FPGA all'Host richiesta [MB/s]
10	70	0,28	2,8%	28
46	250	1	2,17%	21
88	750	3	3,41%	34

Figura 4.5 Possibili valori di PRI e numero di campioni da acquisire a 250MHz

I dati inviati dal convertitore analogico-digitale, vengono memorizzati all'interno di un file. La configurazione utilizzata per l'acquisizione di file di prova è mostrata in Figura 4.6.

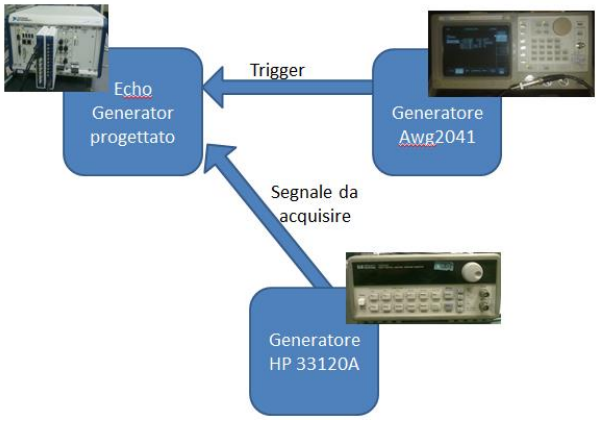

Figura 4.6 Configurazione utilizzata per l'acquisizione di segnali

Una volta scritti, i file possono essere letti, attraverso un altro VI. Questo programma è stato creato per leggere e visualizzare i dati all'interno di un file. In questo programma vengono separate le varie acquisizioni, separati in base alla dimensione dell'acquisizione (presente nel file ausiliario) e visualizzati all'interno di un Waveform Graph. Il Waveform Graph esegue automaticamente una ricostruzione lineare dei campioni, permettendo di visualizzare i segnali acquisiti in maniera continua. Sono stati effettuati dei test in acquisizione con un segnale di prova, una sinusoide creata con un generatore HP 33120A.

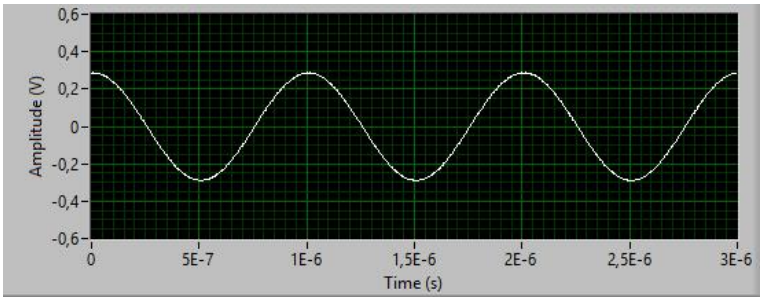

Figura 4.7 Segnale sinuoidale a 1MHz memorizzato su disco

Le prove iniziali di acquisizione sono state eseguite con un segnale sinusoidale a 1MHz e 0.6Vpp. In questo caso, il numero di punti acquisiti per periodo della sinusoide, con una frequenza di campionamento di 250MHz, sono 250 e si riesce a visualizzare completamente la sinusoide. È

presente sempre il generatore AWG2041, per la generazione del segnale di PRI. In questo caso viene utilizzato un periodo di 88µs, corrispondenti ad una finestra di acquisizione di 750 campioni.

Nel grafico in Figura 4.7, l'asse delle ordinate indica l'ampiezza, espressa in volt, del segnale acquisito e l'asse delle ascisse visualizza il tempo, espresso in secondi. Nel programma di lettura dei file, vengono convertiti i numeri interi, provenienti dal convertitore analogico digitale (l'NI 5761 ha un convertitore a 14bit e i valori rappresentati vanno da -8191 a 8191) in ampiezze, espresse in volt, del segnale. Questa operazione è espressa nell'Equazione 4.1.

$$V = \text{campione} * \frac{0{,}6}{8191}$$

Equazione 4.1

Per l'Equazione 4.1, è stato utilizzato il valore massimo di tensione, che il convertitore può acquisire ($1{,}2V_{pp}$), e il valore massimo rappresentabile dal convertitore (8191). Per l'asse delle ordinate, invece, è stata utilizzata la frequenza di campionamento, a 250MHz, per convertire il vettore del numero di campioni in un vettore tempo.

Nella Figura 4.8, sono sovrapposte cinque acquisizioni consecutive del segnale sinusoidale. I parametri del segnale acquisito sono gli stessi della precedente acquisizione.

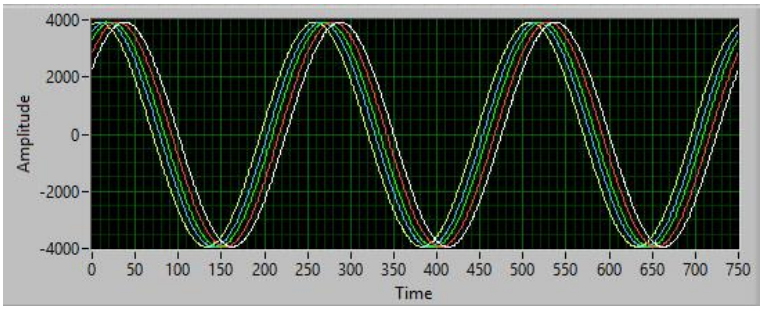

Figura 4.8 Cinque acquisizioni consecutive di un segnale sinusoidale

Nel grafico in Figura 4.8, l'asse delle ordinate visualizza il numero intero del segnale acquisito, in uscita dal convertitore, e l'asse delle ascisse indica il numero progressivo associato ad ogni campione in ordine crescente (con indice 1 il primo campione, con 2 il secondo e così via).

Si può notare che la fase varia lentamente nel segnale acquisito ma questo è indice che il periodo del segnale di PRI (88µs) è prossimo ad un multiplo del periodo del segnale acquisito(1µs).

Visualizzazione dei dati generati su oscilloscopio

I dati generati dall'AWG NI PXIe-5451, possono essere visualizzati con l'uso di un oscilloscopio. Per il corretto funzionamento dell'EG, deve essere fornito il segnale di PRI, per l'avvio delle operazioni di acquisizione e generazione. Questo segnale è creato con l'AWG2041. I trigger generati corrispondono a quelli ricevuti dal radar RDA, ossia con un periodo di 10, 46 e 88µs.

Generazione del PRI

Il PRI è stato generato con l'AWG 2041. Il PRI, nelle tre versioni, è stato generato con forme d'onda arbitrarie. Queste forme d'onda sono state create con l'aiuto dell'editor grafico, presente nel dispositivo. Nella Figura 4.9 e Figura 4.10, vengono mostrati i PRI a 46µs e a 88µs.

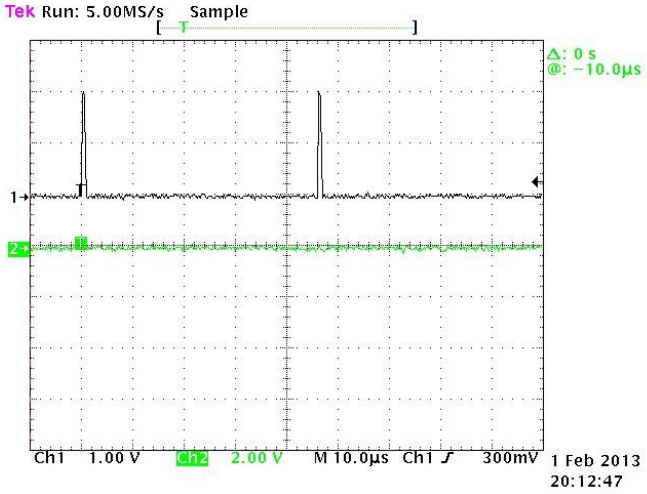

Figura 4.9 Visualizzazione, sull'oscilloscopio, del PRI a 46µs

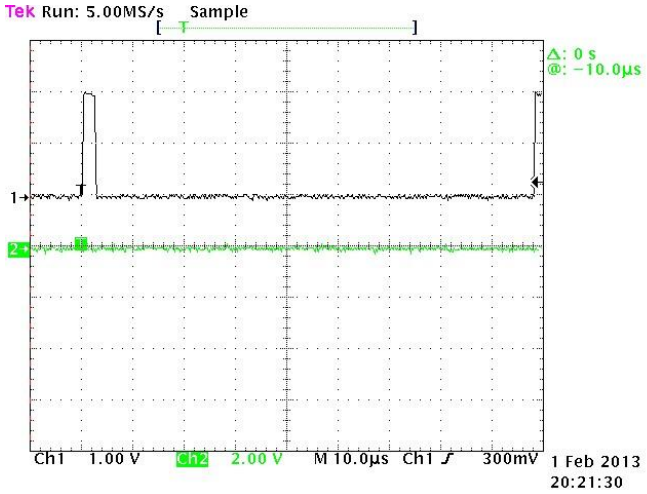

Figura 4.10 Visualizzazione, sull'oscilloscopio, del PRI a 88µs

Confronto tra i dati acquisiti e i dati calcolati

Sono stati simulati, per testare il sistema efficacemente, le situazioni critiche per il radar. I casi più difficili sono a bassa quota, in cui il PRI assume il valore più basso, e ad alta quota, in cui la dimensione dell'impulso è la massima registrabile. Sono riportati, in Figura 4.11, i casi presi in considerazione per testare l'Echo Generator.

	Bassa quota	Bassa quota	Quota media	Alta quota
Coordinata Z [m]	250	400	4500	8500
Angolo di Off-Nadir [rad]	0,8	0,05	0,524	0,05
Periodo di PRI [µs]	10	10	46	88
Dimensione dell'eco calcolato	94	104	1354	2218
Durata dell'eco generato a 200MHz [µs]	0,47	0,52	6,77	11,09

Figura 4.11 Dati cinematici utilizzati per la generazione degli echi riflessi

Viene mostrato in Figura 4.12, la configurazione per acquisire le immagini con l'oscilloscopio. Il generatore AWG2041 crea il segnale di trigger e l'oscilloscopio, a quattro canali, permette di visualizzare sia il segnale di trigger che il segnale generato.

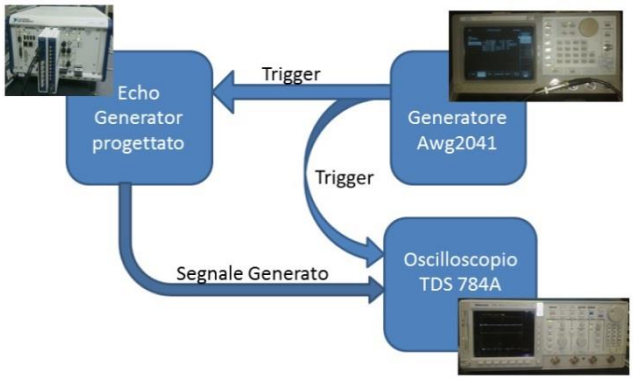

Figura 4.12 Configurazione utilizzata per la visualizzazione dei dati sull'oscilloscopio

Nel primo caso a bassa quota (250 m), la dimensione dell'impulso riflesso è di soli 0,47µs. Nella Figura 4.13, è mostrato l'eco calcolato su FPGA, che sarà generato. Per visualizzare il segnale sull'oscilloscopio, si è preferito generare gli impulsi, non con la potenza reale, ma ai valori di tensione massimi permessi dal sistema. Questo può essere implementato facilmente, sostituendo l'algoritmo per il calcolo delle componenti I&Q, con una moltiplicazione. In questo modo l'eco normalizzato, calcolato nel modello, sarà solo moltiplicato per il valore massimo rappresentato con il tipo Integer 16, quello generato con l'AWG. Quindi, il segnale generato è stato riportato al valore massimo di output. Un altro parametro per gestire l'amplificazione del segnale, è la variabile Gain, definita in fase di configurazione dell'AWG. Questo parametro rappresenta la tensione massima generata in uscita dall'AWG. In questo caso, Gain è stato fissato a 2,5V.

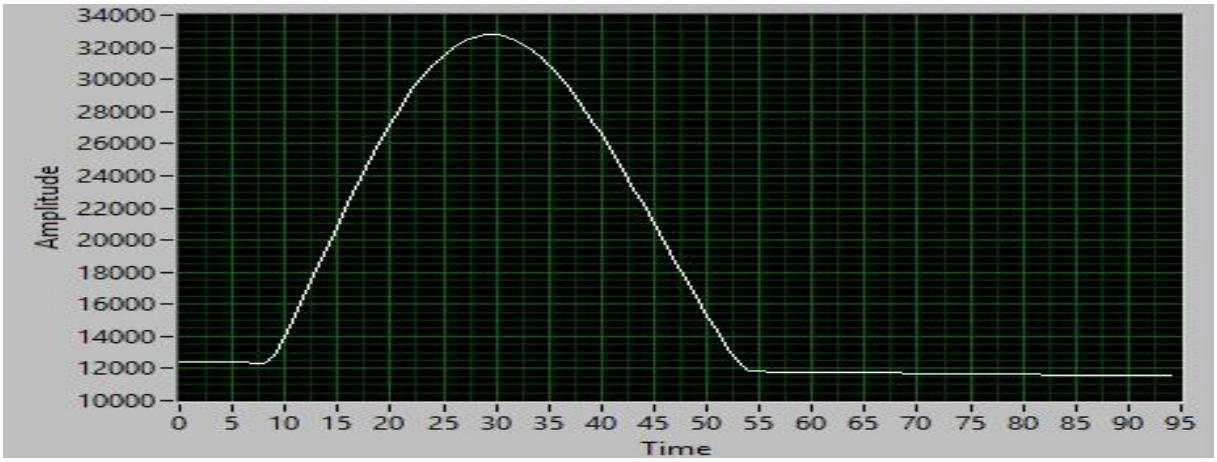

Figura 4.13 Segnale calcolato su FPGA per la generazione, nel caso di 250m e 0,8rad

Sono mostrati in Figura 4.14 e Figura 4.15, i segnali campionati con l'oscilloscopio. Nella Figura 4.14, è mostrato il caso in cui viene trascurato il ritardo, calcolato in base alla quota. In questo caso il segnale viene generato, quando si riceve il segnale di PRI. Nella Figura 4.14 è stato acquisito su Channel1, con il colore nero, il segnale di trigger e con Channel2, con colore verde, il segnale generato. Questa configurazione è ripetuta in tutte le immagini successive, acquisite dall'oscilloscopio.

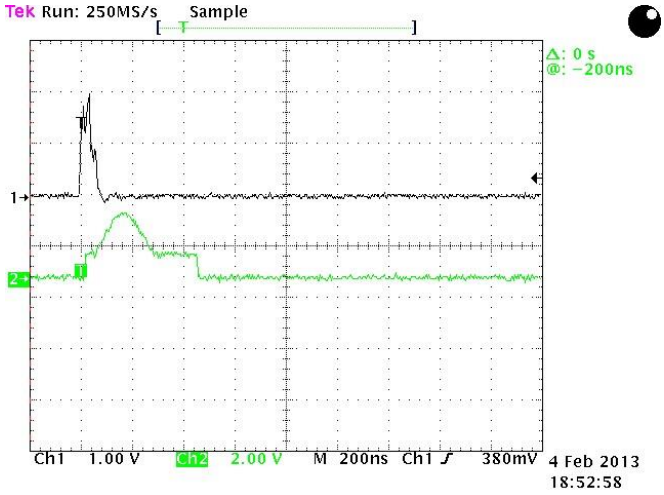

Figura 4.14 Segnale acquisito con l'oscilloscopio, senza ritardo dal PRI

Si può notare dalla Figura 4.14, che la durata dell'eco generato coincide con quella stimata in Figura 4.11. Nella Figura 4.15, viene considerato il ritardo di generazione, quantificabile attraverso l'Equazione 4.2.

$$t_{ric} = 2\frac{H}{c} = 2\frac{Z}{\mathrm{c} * \cos\xi}$$

Equazione 4.2

In questo caso, il ritardo stimato è di 1µs. In Figura 4.15, viene verificato il ritardo, che coincide con il valore precedente.

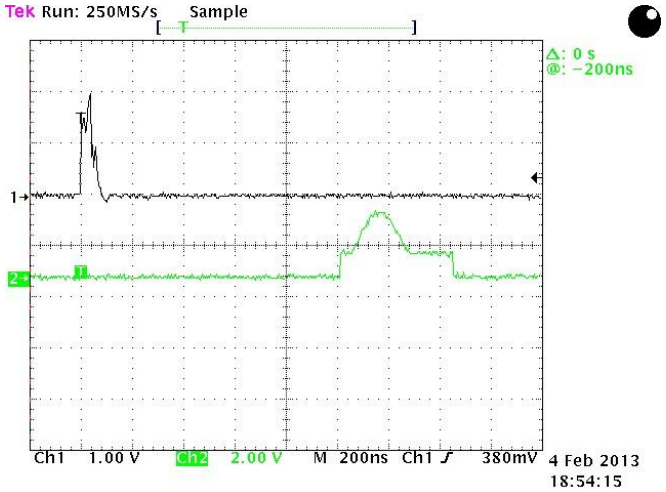

Figura 4.15 Segnale acquisito con l'oscilloscopio, con ritardo reale

Nel secondo caso a bassa quota (400m), la dimensione dell'impulso riflesso è di 0,52µs. Nella Figura 4.16, è mostrato l'eco calcolato su FPGA, che sarà generato. Anche in questo caso, i valori dei campioni sono stati riportati al valore massimo.

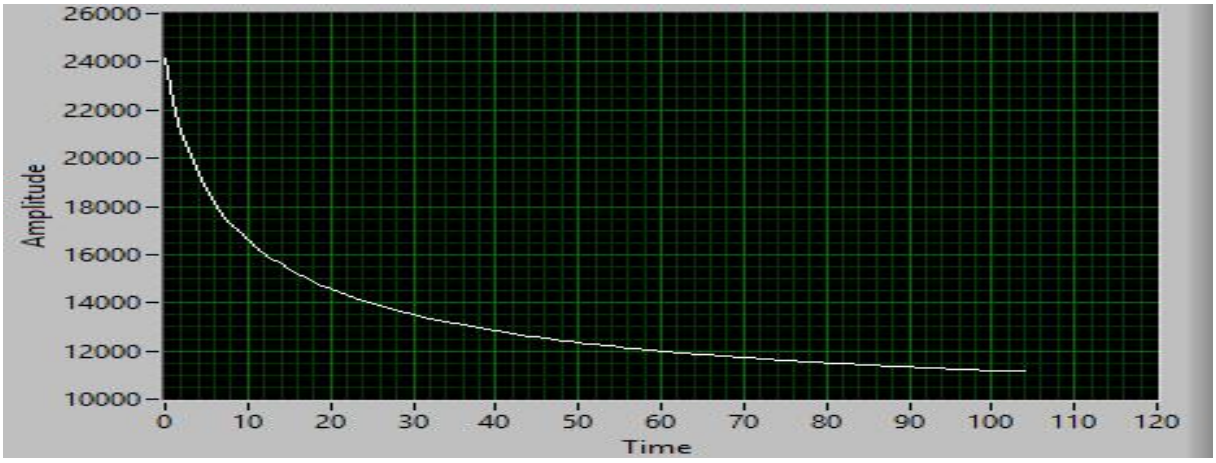

Figura 4.16 Segnale calcolato su FPGA per la generazione, nel caso di 400m e 0,05rad

Sono mostrati in Figura 4.17 e Figura 4.18, i segnali campionati con l'oscilloscopio. Nella Figura 4.17, è mostrato il caso in cui viene trascurato il ritardo, calcolato in base alla quota. In questo caso il segnale viene generato, quando si riceve il segnale di PRI.

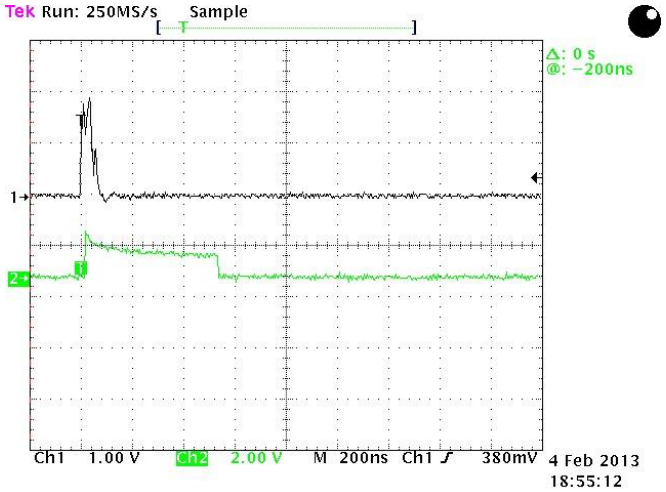

Figura 4.17 Segnale acquisito con l'oscilloscopio, senza ritardo dal PRI

Si può notare dalla Figura 4.17, che la durata dell'eco generato coincide con quella stimata in Figura 4.11. Nella Figura 4.18, viene considerato il ritardo di generazione, quantificabile attraverso l'Equazione 4.2.

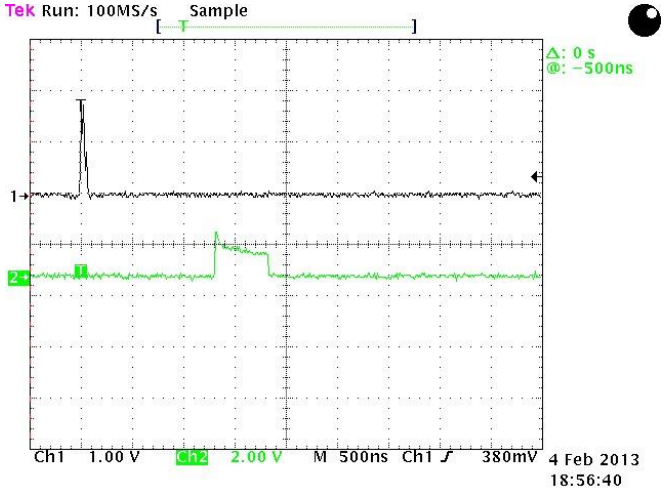

Figura 4.18 Segnale acquisito con l'oscilloscopio, con ritardo reale

In questo caso, il ritardo stimato è di 1,3μs. In Figura 4.18, viene verificato il ritardo, che coincide con il valore precedente.

A media quota (4500m), la dimensione dell'impulso riflesso è di 6,77μs. Nella Figura 4.19, è mostrato l'eco calcolato su FPGA, che sarà generato. Anche in questo caso, i valori dei campioni sono stati riportati al valore massimo.

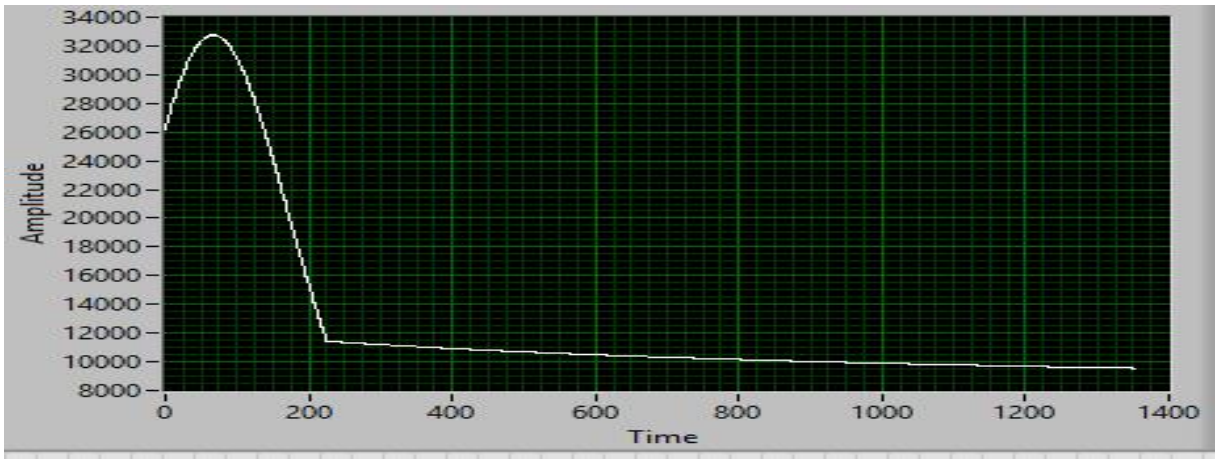

Figura 4.19 Segnale calcolato su FPGA per la generazione, nel caso di 4500m e 0,524rad

Sono mostrati in Figura 4.20 e Figura 4.21, i segnali campionati con l'oscilloscopio. Nella Figura 4.14, è mostrato il caso in cui viene trascurato il ritardo, calcolato in base alla quota. In questo caso il segnale viene generato, quando si riceve il segnale di PRI.

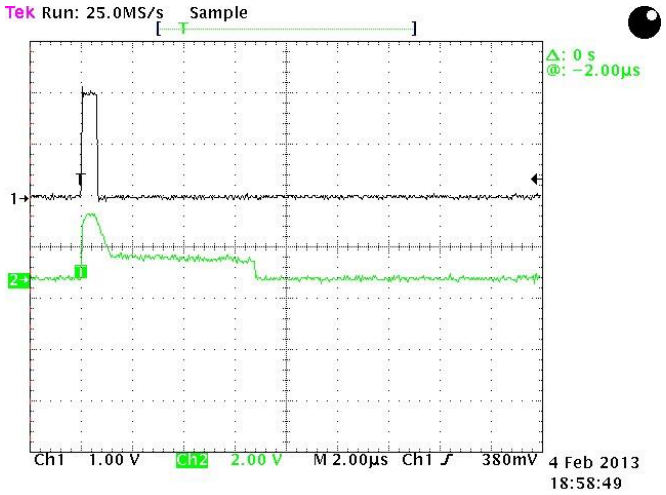

Figura 4.20 Segnale acquisito con l'oscilloscopio, senza ritardo dal PRI

Si può notare dalla Figura 4.20 che la durata dell'eco generato coincide con quella stimata in Figura 4.11. Nella Figura 4.21, viene considerato il ritardo di generazione, quantificabile attraverso l'Equazione 4.2.

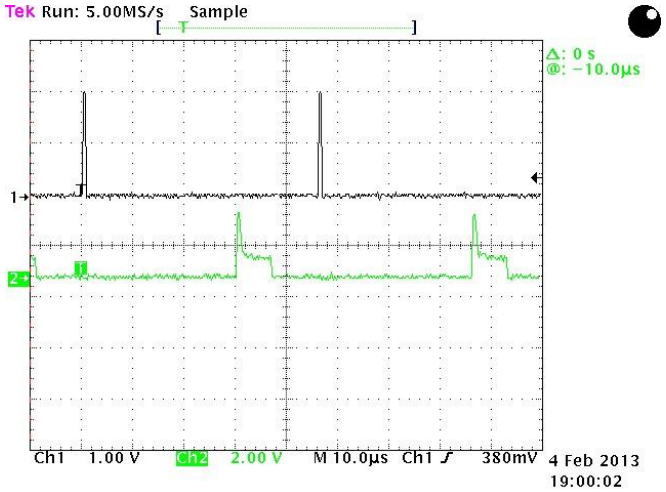

Figura 4.21 Segnale acquisito con l'oscilloscopio, con ritardo reale

In questo caso, il ritardo stimato è di 30µs. In Figura 4.21, viene verificato il ritardo, che coincide con il valore precedente.

In alta quota (8500m), la dimensione dell'impulso riflesso è di 11,09µs. Nella Figura 4.22, è mostrato l'eco calcolato su FPGA, che sarà generato. Anche in questo caso, i valori dei campioni sono stati riportati al valore massimo.

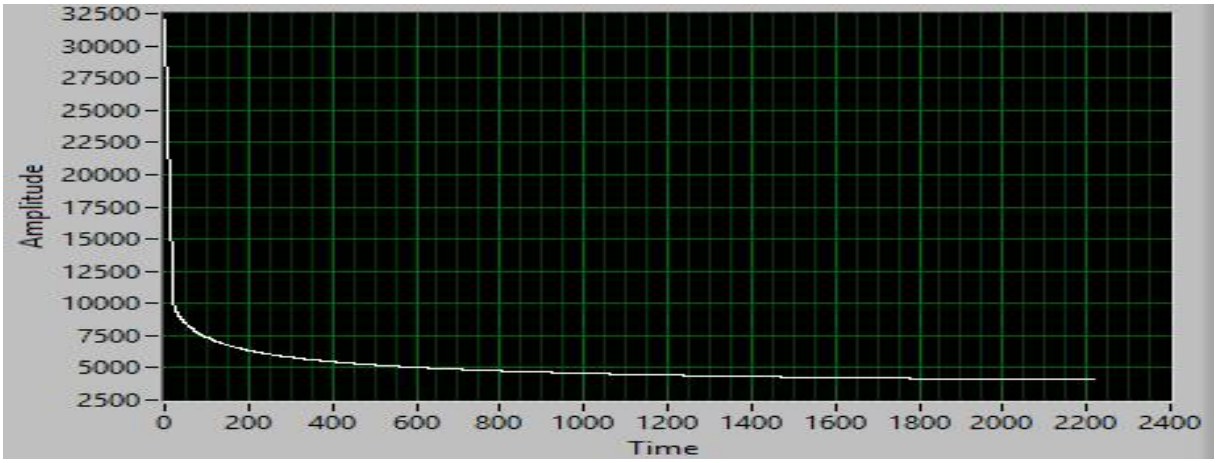

Figura 4.22 Segnale calcolato su FPGA per la generazione, nel caso di 8500m e 0,05rad

Sono mostrati in Figura 4.23e Figura 4.24, i segnali campionati con l'oscilloscopio. Nella Figura 4.23, è mostrato il caso in cui viene trascurato il ritardo, calcolato in base alla quota. In questo caso il segnale viene generato, quando si riceve il segnale di PRI.

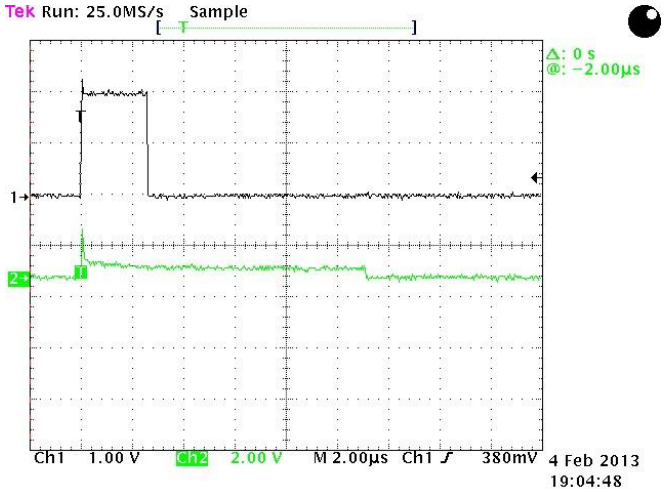

Figura 4.23 Segnale acquisito con l'oscilloscopio, senza ritardo dal PRI

Si può notare dalla Figura 4.23, che la durata dell'eco generato coincide con quella stimata in Figura 4.11. Nella seconda immagine, viene considerato il ritardo di generazione, quantificabile attraverso l'Equazione 4.2.

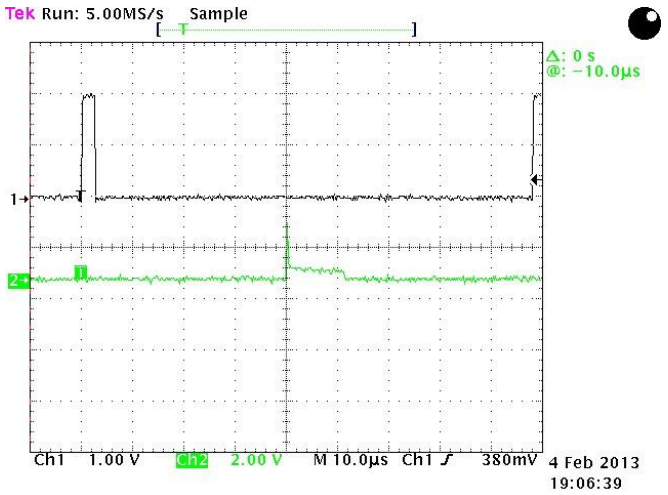

Figura 4.24 Segnale acquisito con l'oscilloscopio, con ritardo reale

In questo caso, il ritardo stimato è di 40µs. In Figura 4.24, viene verificato il ritardo, che coincide con il valore stimato.
Nelle varie immagini acquisite con l'oscilloscopio, è stato verificato il corretto funzionamento dell'EG.

Acquisizione dei segnali generati con sistema chiuso ad anello

L'ultima prova eseguita sul sistema, per verificare il corretto funzionamento dell'EG, è l'acquisizione del segnale generato dall'AWG. Nel normale funzionamento, non è possibile acquisire il segnale generato, poiché l'acquisizione e la generazione sono inserite in due finestre temporali differenti. L'acquisizione corrisponde all'impulso inviato, ossia immediatamente dopo la ricezione del PRI, invece la generazione al segnale ricevuto, che arriva al radar con un ritardo dal

PRI. Quindi, per questa prova, è stata utilizzata una versione modificata del programma, che permette di generare il segnale senza considerare il ritardo, non appena viene ricevuto il segnale di PRI. La potenza del segnale generato, inoltre, è stata adattata alla tensione massima valutabile dall'adapter module NI5761, di 0,6V. Anche in questo caso, nelle acquisizioni è stata utilizzata una frequenza di campionamento di 250MHz, la massima utilizzabile dall'NI 5761. Una rappresentazione del sistema chiuso ad anello viene mostrata in Figura 4.25.

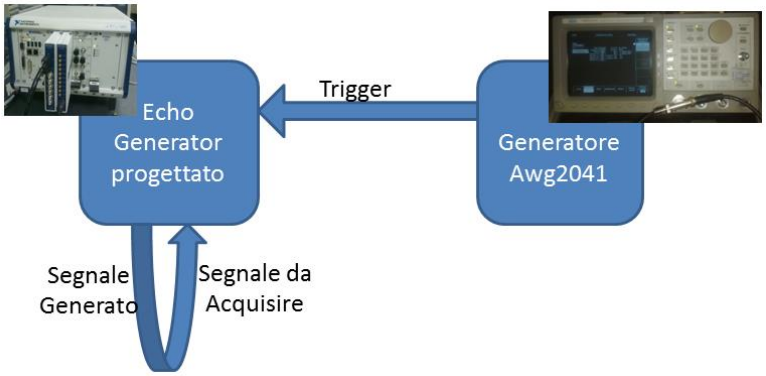

Figura 4.25 Configurazione utilizzata per la visualizzazione dei dati sull'oscilloscopio

Sono state eseguite varie prove per le situazioni critiche dell'Echo Generator. I casi più difficili sono a bassa quota, in cui il PRI assume il valore più basso, e ad alta quota, in cui la dimensione dell'impulso è la massima registrabile. I casi considerati sono gli stessi utilizzati per visualizzare i segnali sull'oscilloscopio, riportati in Figura 4.11.

Nel primo caso in bassa quota (250m), il PRI ricevuto dal radar è di 10μs. La finestra di acquisizione corrispondente è di 70 campioni. Questo coincide ad un tempo di acquisizione di 0,28μs. I dati memorizzati vengono mostrati in Figura 4.26.

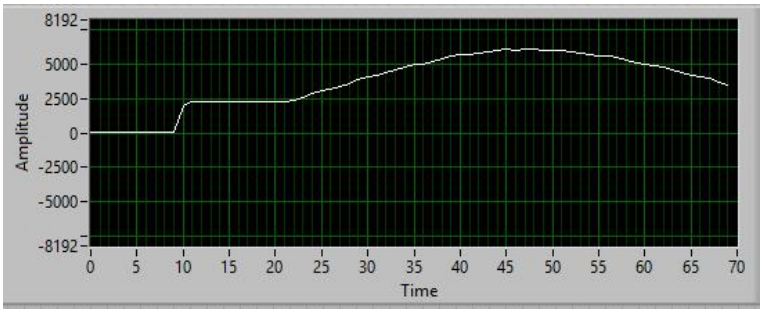

Figura 4.26 Segnale acquisito nel caso di 250m e 0,8rad con una finestra di 70 campioni

Nel caso di 250m e 0,8rad, la durata del segnale è di 0,47μs. Quindi non viene memorizzato l'intero segnale. Per visualizzare l'intero segnale, è stata modificata la finestra di acquisizione in 250 campioni, corrispondenti a 1μs. Con una finestra di acquisizione maggiore della durata del segnale, è possibile memorizzare l'intero segnale generato. In questo modo, si porta il sistema ad una velocità maggiore di trasferimento dei dati e di scrittura su disco. L'aumento, fino a 95MB/s (per 2 canali a 16bit e periodo di 10μs), del datarate medio verso disco può essere supportato, solo per un breve periodo. Fin quando la coda FIFO DMA non mostra una condizione di overflow, è possibile continuare l'esecuzione. I dati acquisiti, prima che avvenga una condizione di errore, sono mostrati in Figura 4.27.

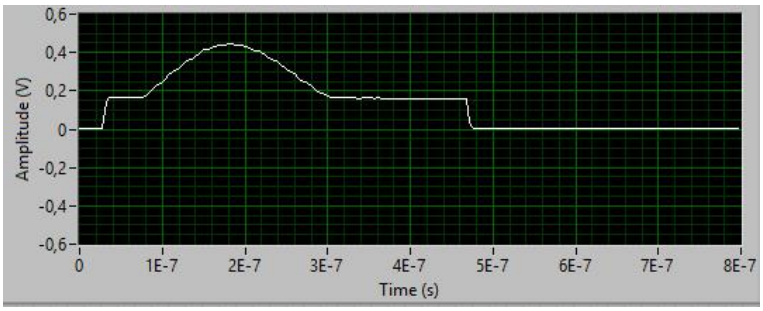

Figura 4.27 Segnale acquisito nel caso di 250m e 0,8rad con una finestra di 250 campioni

Nel secondo caso in bassa quota (400m), il PRI ricevuto dal radar è di 10µs. La finestra di acquisizione corrispondente è di 70 campioni. Questo coincide ad un tempo di acquisizione di 0,28µs. i dati memorizzati vengono mostrati in Figura 4.28.

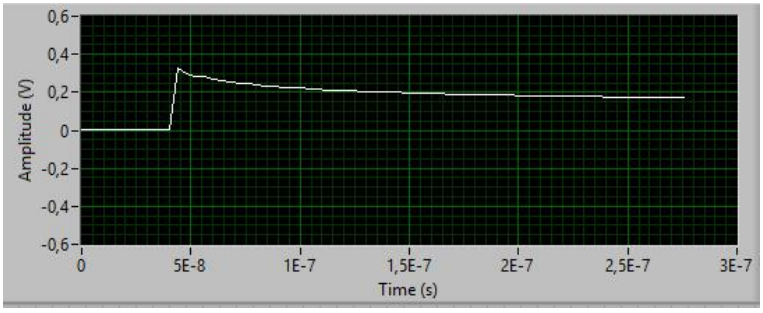

Figura 4.28 Segnale acquisito nel caso di 400m e 0,05rad con una finestra di 70 campioni

Nel caso di 400m e 0,05rad, la durata del segnale è di 0,52µs. Anche in questo caso non viene memorizzato l'intero segnale. Per visualizzare l'intero segnale, è stata modificata la finestra di acquisizione in 250 campioni, corrispondenti a 1µs. Le considerazioni fatte nel caso precedente, sul datarate medio verso disco, rimangono uguali, perché è stata utilizzata sempre una finestra di 250 campioni. I dati acquisiti, prima che avvenga una condizione di overflow sulla coda FIFO DMA, sono mostrati in Figura 4.29.

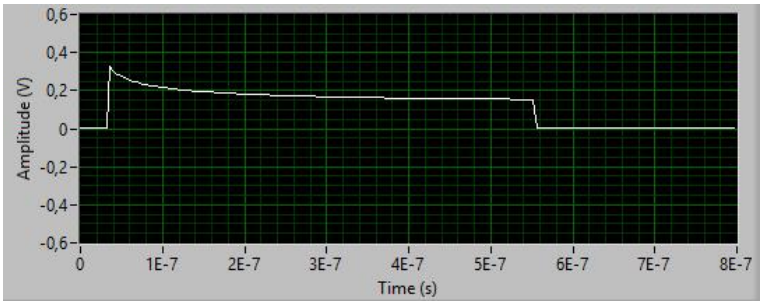

Figura 4.29 Segnale acquisito nel caso di 400m e 0,05rad con una finestra di 250 campioni

Nel caso di quota media (4500m), il PRI ricevuto dal radar è di 46µs. La finestra di acquisizione corrispondente è di 250 campioni. Questo coincide ad un tempo di acquisizione di 1µs. i dati memorizzati vengono mostrati in Figura 4.30.

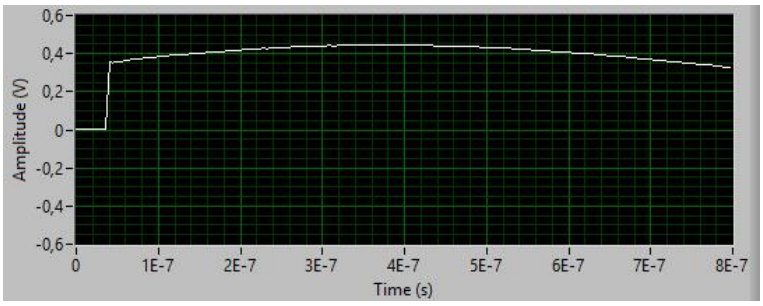

Figura 4.30 Segnale acquisito nel caso di 4500m e 0,524rad con una finestra di 250 campioni

Nel caso di 4500m e 0,524rad, la durata del segnale è di 6,77µs. Anche in questo caso, non viene memorizzato l'intero segnale. Per visualizzare l'intero segnale, è stata modificata la finestra di acquisizione in 2000 campioni, corrispondenti a 8µs. Le considerazioni fatte nel primo caso, sul datarate medio verso disco, rimangono simili. In questo caso, con un PRI di 46µs e 2000 campioni acquisiti, si arriva a 166MB/s. Quindi, la condizione di overflow accade più velocemente. I dati acquisiti, prima che avvenga una condizione di overflow sulla coda FIFO DMA, sono mostrati in Figura 4.31.

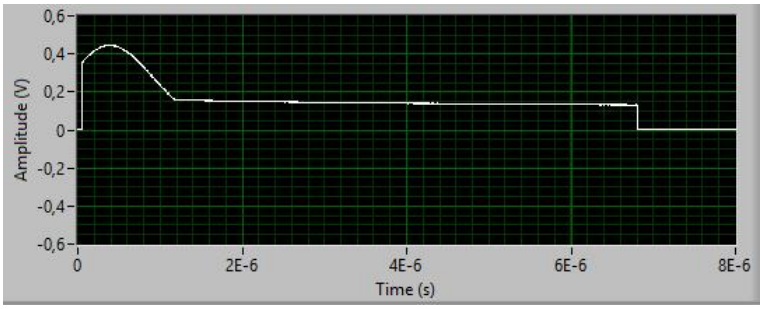

Figura 4.31 Segnale acquisito nel caso di 4500m e 0,524rad con una finestra di 2000 campioni

Nel caso ad alta quota (8500m), il PRI ricevuto dal radar è di 88µs. La finestra di acquisizione corrispondente è di 750 campioni. Questo coincide ad un tempo di acquisizione di 3µs. i dati memorizzati vengono mostrati in Figura 4.32.

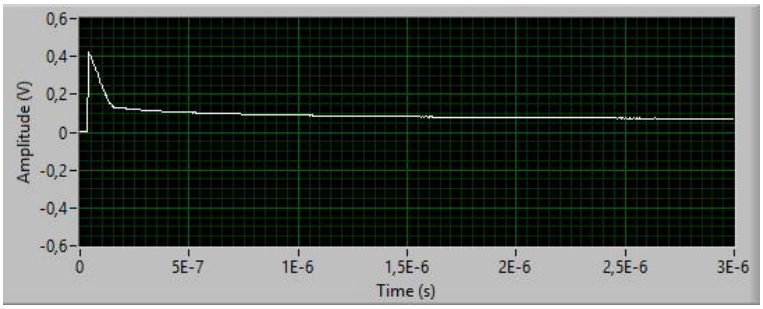

Figura 4.32 Segnale acquisito nel caso di 8500m e 0,05rad con una finestra di 750 campioni

Nel caso di 8500m e 0,05rad, la durata del segnale è di 11,09µs. Anche in questo caso, non viene memorizzato l'intero segnale. Per visualizzare l'intero segnale, è stata modificata la finestra di acquisizione in 3000 campioni, corrispondenti a 12µs. Le considerazioni fatte nel primo caso, sul datarate medio verso disco, rimangono simili. In questo caso, con un PRI di 88µs e 3000 campioni acquisiti, si arriva a 130MB/s. Quindi la condizione di overflow accade più velocemente. I dati

acquisiti, prima che avvenga una condizione di overflow sulla coda FIFO DMA, sono mostrati in Figura 4.33.

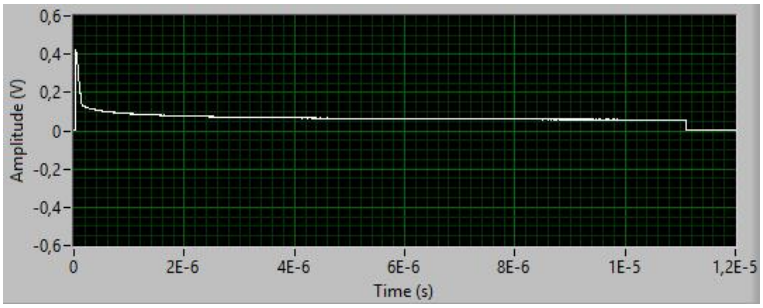

Figura 4.33 Segnale acquisito nel caso di 8500m e 0,05rad con una finestra di 3000 campioni

Se la catena di generazione e di acquisizione funziona correttamente, i segnali acquisiti devono corrispondere ai segnali calcolati su FPGA I dati visualizzati, in questo paragrafo, corrispondono ai segnali mostrati in Figura 4.13, Figura 4.16, Figura 4.19 e Figura 4.22. In queste immagini sono mostrati gli echi calcolati direttamente su FPGA. Anche in questo test, l'EG ha mostrato una risposta corretta.

Conclusioni

Obiettivo del nostro lavoro era la creazione di un Echo Generator, che permettesse di simulare gli echi provenienti dalla superficie di Marte e porli in ingresso al Radar Doppler Altimeter (RDA) di EXOMARS, in modo da riuscire a testare tale strumento in varie condizioni operative.

Attraverso prove sperimentali, è stato possibile valutare il corretto funzionamento dell'eco generator, di cui si sono voluti valutare la temporizzazione, la capacità di trasferimento dei campioni acquisiti, la velocità nell'elaborazione di nuovi echi simulati e le prestazioni in fase di generazione del segnale analogico.

I test effettuati hanno permesso di appurare che tale sistema rispondeva in maniera coerente a tutti i requisiti inizialmente richiesti.

Di notevole interesse, nel corso del lavoro, è stato lo studio del sistema hardware, delle sue caratteristiche, del linguaggio di programmazione Labview e Labview FPGA.

Altrettanto stimolante è stata l'opportunità di poter usufruire degli appositi strumenti di misura disponibili in un laboratorio elettronico come oscilloscopi, multimetri e generatori di segnali.

A nostro modesto avviso, il sistema potrebbe prestarsi ad ulteriori sviluppi utilizzando un modulo adapter, per la fase di generazione, con una frequenza più elevata. Ciò potrebbe consentire l'eliminazione dell'AWG PXIe-5451, semplificando notevolmente la programmazione e aumentando l'affidabilità dell'Echo Generator.

Riferimenti

Molti documenti sono stati presi dal sito National Instruments, http://www.ni.com/it/ , per la grande disponibilità di materiale dell'hardware e del software proprietario.

A Waveform Model for Near-nadir Radar Altimetry applied to the Cassini Mission to Titan; Giovanni Alberti, Luca Festa, Claudio Papa, and Guido Vingione; on IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, VOL. 47, NO. 7, JULY 2009

Digital Signal Processing System-Level Design Using LabVIEW; Nasser Kehtarnavaz e Namjin Kim; Elsevier 2005

LabView Programming, Data Acquisition and Analysis; Jeffrey Y. Beyon; Prentice-Hall 2001

LabView Advanced Programming Techniques; Rick Bitter, Taqi Mohiuddin, Matt Nawrocki; CRC Press 2001

Handbook Of Signal Processing Systems; S. Bhattacharyya, Ed F. Deprettere, Rainer Leupers, Jarmo Takala; Springer 2010

Labview Graphical Programming; Gary W. Johnson , Richard Jennings; McGraw-Hill 2006

Exploiting Formation Flying for Earth Science: P-band Distributed Synthetic Aperture Radar; Giovanni Alberti, Giancarmine Fasano, Marco D'Errico, Stefano Cesare, Gianfranco Sechi, Massimiliano Marcozzi, Leonardo Mazzini, Andrea Torre, Mario Cosmo, Roberto Formaro, Quirino Rioli
Disponibile online http://www.corista.unina.it/Docs/exploiting_formation.pdf

Segnali Complessi: modulazione in fase e quadratura;
Disponibile Online
http://www-dsp.elet.polimi.it/fdt/prati/LEZIONI/06_modulazione_IQ/Modulazione_IQ.PDF

Fundamentals of Remote Sensing - 2nd Edition; George Joseph, Orient Blackswan; Universities Press 2005

Telerilevamento attivo a microonde: il radar altimetro; Roberto Seu
Disponibile Online
http://infocom.uniroma1.it/~robseu/Sistemi%20Radar%20Spaziali/ALTIMETRO.ppt

Delay-Doppler (or SAR) Altimetry;
Disponibile Online http://www.altimetry.info/html/alti/principle/alti_doppler_en.html

Satellite Altimetry, Satellite altimetry and Earth sciences; L.L. Fu and A. Cazenave Ed, D.B., J.C. Ries, B.J. Haines, L.L. Fu, P.S. Callahan; Academic Press 2001

ExoMars EDL Demonstrator Module (EDM) Mission and Design Overview; O. Bayle, L. Lorenzoni, Th. Blancquaert, S. Langlois, Th. Walloschek, S. Portigliotti, M.Capuano
Disponibile Online
http://www.planetaryprobe.org/sessionfiles/session2/presentations/2_Bayle_ExoMars_EDM_Overview.pdf

ExoMars RDA EGSE Echo Simulator System (ESS);
Disponibile Online http://www.corista.eu/exomars.html

The ExoMars Entry, Descent and Landing Demonstrator Module (EDM);
Disponibile Online http://exploration.esa.int/science-e/www/object/index.cfm?fobjectid=47852

Che cos'è PXI?; National Instruments 2011
Disponibile online http://zone.ni.com/devzone/cda/tut/p/id/8650

Virtex-5 Family Overview; Xilinx 2009
Disponibile online http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf

Virtex-5 FPGA User Guide; Xilinx 2010
Disponibile online http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

Virtex-5 FPGA RocketIO GTP Transceiver User Guide; Xilinx 2009
Disponibile online http://www.xilinx.com/support/documentation/user_guides/ug196.pdf

Virtex-5 FPGA Integrated Endpoint Block for PCI Express Designs User Guide; Xilinx 2009
Disponibile online http://www.xilinx.com/support/documentation/user_guides/ug197.pdf

Virtex-5 FPGA XtremeDSP Design Considerations User Guide; Xilinx 2010
Disponibile online http://www.xilinx.com/support/documentation/user_guides/ug193.pdf

Data Sheet NI PXI-8133 Embedded Controller for PXI Express; National Instruments
Disponibile online http://sine.ni.com/ds/app/doc/p/id/ds-267/lang/it

User Manual and Specifications NI PXIe-8133; National Instruments
Disponibile online
http://digital.ni.com/manuals.nsf/websearch/BC9F1203CE7D41C886257745004C97BB

Data Sheet FlexRio FPGA; National Instruments
Disponibile online http://www.ni.com/pdf/products/us/cat_flexriofpga.pdf

NI FlexRIO FPGA Module specification; National Instruments
Disponibile online
http://digital.ni.com/manuals.nsf/websearch/BE37162D3EDFCE9C86257662007077CC

Data Sheet PXIe-1082; National Instruments
Disponibile online http://www.ni.com/pdf/products/us/cat_pxie1082.pdf

Data Sheet NI PXIe-1082 Backplane; National Instruments
Disponibile online http://sine.ni.com/nips/cds/view/p/lang/it/nid/208082#

NI pxie-1082 User Manual; National Instruments
Disponibile online
http://digital.ni.com/manuals.nsf/websearch/B8B44E204AE123FB862576B80059848E

Data Sheet NI 5781R; National Instruments; National Instruments
Disponibile online http://sine.ni.com/ds/app/doc/p/id/ds-212/lang/it

NI 5781R User Guide and Specifications; National Instruments
Disponibile online
http://digital.ni.com/manuals.nsf/websearch/841B36C8127278AC862576B300771272

NI 5761R User Guide and Specifications; National Instruments
Disponibile online www.**ni**.com/pdf/manuals/375509a.pdf

NI PXIe-5451 Specifications; National Instruments
Disponibile online www.ni.com/pdf/manuals/372936a.pdf

FPGA Design, Development and Programming Tutorial; National Instruments
Disponibile online ftp://ftp.ni.com/pub/devzone/pdf/tut_3358.pdf

Introduzione all'utilizzo di NI CompactRIO; Gabriele Melani
Disponibile online http://www.slideshare.net/ale914/introduzione-allutilizzo-del-compact-rio

Data Sheet Labview FPGA Module; National Instruments
Disponibile online http://www.ni.com/pdf/products/us/labview_fpga_module.pdf

Help Labview FPGA Module; National Instruments
Disponibile online
http://digital.ni.com/manuals.nsf/websearch/BB494728D829D96F86257750007E7068

Il compilatore LabVIEW; National Instruments
Disponibile online http://zone.ni.com/devzone/cda/tut/p/id/11920

Labview Fpga Training Slides; National Instruments
Disponibile online ftp://ftp.ni.com/pub/devzone/tut/fpga_training_slides.zip

Alcuni documenti, utilizzati per la definizione delle specifiche, sono riservati al consorzio CO.RI.S.T.A. e all'ESA. Pertanto, saranno riportati solo i titoli descrittivi dei documenti.

Exomars Mission Objectives Document

Radar Doppler Unit Requirements Specification

Echo Simulator System Requirement Specification

RDA Echo Generator Detailed Design Definition

RDA Real Time Model Procurement Requirements Specification

RDA Real Time Model Architectural Design Description

Ringraziamenti

Giunto alla fine di questo lungo percorso, desidero ringraziare tutte le persone che hanno contribuito al raggiungimento di questo traguardo, così importante della mia vita, ma ricordare e ringraziare tutti quelli che mi hanno sostenuto, in queste poche righe, è una cosa molto difficile.

Desidero ringraziare il Prof. Davide De Caro, per gli insegnamenti e il tempo dedicato all'opera di sviluppo e di scrittura della tesi. Inoltre ringrazio in particolar modo, l'Ing. Luca Ciofaniello che ha messo a mia disposizione la sua professionalità, seguendo con scrupolo e competenza il lavoro presentato nel testo e che mi ha fornito consigli sempre ottimi.

Un grazie particolare deve essere rivolto a Massimo Gagliardi, Gianni Alberti, Gianfranco Palmese e Dario Califano, con cui ho collaborato ottimamente e che si sono dimostrati sempre disponibili nei miei confronti. Grazie a tutto lo staff del consorzio Co.Ri.S.T.A., in particolare a: Claudio, Peppe, Giulia, Maria Rosaria e Cristina.

Un ringraziamento va sicuramente alla mia famiglia che mi ha sempre sostenuto, aiutato con vari consigli, incoraggiato per realizzare i miei obiettivi. Quando ne avevo bisogno, erano sempre con me, confermandosi un punto di riferimento essenziale nella mia vita.

Un grazie infinito a Dafne che mi è stata sempre vicina, con cui ho passato dei momenti unici e che mi ha dato la forza per andare avanti (ne è servita molta), per raggiungere i miei traguardi. È stata sempre al mio fianco in questo cammino.

Non posso dimenticare gli amici Tommaso, Gianpaolo, Pasquale, Teresa, Alessandro, Nunzio, Roberto, Luca e Salvatore, che mi hanno accompagnato in questo viaggio e con cui ho passato molti momenti speciali, sia di svago che di "studio" (sono d'obbligo le virgolette).

Inoltre vorrei ringraziare i miei amici, oltre che coinquilini Alessandro, Salvatore, Stefano, Francesco, e Giuseppe per le partite a FIFA, i pranzi con le schede e il tempo passato con allegria tra una studiata e un'altra.