UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Tecnica e Gestione dei Sistemi Industriali

Corso di Laurea Triennale in Ingegneria Meccanica e Meccatronica

Tesi di Laurea

Simulazione Real-Time nell'elettronica di potenza e ambiente di sviluppo Xilinx System Generator

Relatore:

Prof. PAOLO MATTAVELLI

Laureando:

ALESSANDRO TOLIN

ANNO ACCADEMICO 2015-2016

Alla mia famiglia

Sommario

Nel campo dell'elettronica, dei sistemi elettrici, e non solo, durante la fase di prototipazione di un sistema servono delle simulazioni che permettano di fare debug del codice creato e di testare i risultati ottenuti come se il sistema stesse lavorando in un ambiente reale. In questo elaborato si parlerà di Simulazione real-time, essendo uno dei metodi più utilizzati nei sistemi HIL e di come esso è implementato nell'elettronica di potenza. Si andrà a vedere i tre modelli principali della simulazione real-time: continuo, discreto ed ibrido. Per poi introdurre l'ambiente Xilinx System Generator, un ambiente di sviluppo che permette di progettare la modellistica per le simulazioni real-time ibride. Si cercherà di capire com'è strutturato e verranno riportati alcuni esempi nell'ambito dell'elettronica di potenza.

Indice Pag.

Elenco Figure Pag.

Introduzione

Una fase critica nella progettazione di un nuovo sistema, di qualunque tipo, è la fase di test, indispensabile per la rilevazione di malfunzionamenti e difetti.
Generalmente testare un sistema vuol dire inserirlo nel suo ambiente di lavoro e vedere se le funzionalità e le prestazioni sono conformi alle specifiche. Ciò comporta che per poter effettuare un test è necessario attendere che la progettazione e la realizzazione di ogni sua parte sia stata portata a termine; inoltre bisogna assicurarsi di poter effettuare prove senza rischio di danneggiare cose o persone.
Se si pensa ai sistemi elettronici di potenza bisogna tener presente che la loro complessità sta subendo un incremento notevole. Si parla di complessità relativamente ai dispositivi elettronici dei sistemi e ai modelli di controllo da realizzare, sempre più avanzati. È sufficiente pensare ai modelli di controllo digitali che possono essere implementati in ben oltre centomila linee di codice per progetto.
Il test dei sistemi presenta inoltre aspetti peculiari quando si considera la necessità di validare e di verificare tali sistemi, quando questi sono impiegati all'interno di un sistema più generale e complesso. Molto spesso tali problematiche sono vincolate dalla necessità di testare i modelli usando simulazioni real-time, poiché non si ha a disposizione l'effettivo impianto reale o perché la complessità delle prove è tale per cui le prove risultano molto onerose in termini di:

- *Rischi*, perdita vite umane o di capitale;
- *Capitale*, il test nel sistema finito può essere, proibitivamente, dispendioso;
- *Disponibilità*, il sistema o l'ambiente di lavoro non sono disponibili;
- *Sicurezza*, non tutti gli stati del test possono essere raggiunti durante operazioni regolari.

Le presenti considerazioni portano a capire perché al giorno d'oggi occorre dare sempre di più importanza all'integrazione, alla verifica e al test del sistema attraverso la *simulazione real-time*.
Nel corrente elaborato si va a trattare il concetto di simulazione real-time, e la sua interazione con hardware-in-the-loop(HIL), nel quale verranno ripotati alcune soluzioni HIL per l'elettronica di potenza. Si vedrà come la simulazione si è ramificata attraverso prima due modelli classici: continuo e discreto, e le loro differenze; e il modello ibrido che riesce ad inglobare in esso i due modelli.

Si proseguirà col parlare di un ambiente di sviluppo per HIL, Xilinx System Generator, nato in questi ultimi anni e che grazie alla capacità di poter lavorare in ambiente ibrido ed interagendo con MATLAB-Simulink, è riuscito notevolmente a semplificare il modo di simulare e nello stesso tempo di modificare e aggiustare il codice che può essere caricato negli FPGA, microcontrollori e DSP. Infatti questi dispositivi vengono molto spesso utilizzati nella fase di prototipazione di un sistema.

Inoltre verranno riportati degli esempi di schemi a blocchi di semplici dispositivi dell'elettronica di potenza (filtri del primo ordine ed inverter), ottenuti da Xilinx System Generator, in cui a livello di simulazione software si vorrà ottenere i componenti di modellistica indispensabili per una simulazione real-time.

Capitolo 1

Simulazione Real-time

Convenzionalmente lo sviluppo di un sistema elettronico, e non solo, include due fasi: la fase di simulazione off-line, che comprende lo sviluppo a livello software del sistema, e la fase di prototipazione. In questo tipo di processo, il collaudo dell'intero sistema in vari condizioni è possibile solo sul prototipo. Se i risultati dei test funzionali non sono sodisfatti, ciò potrebbe richiedere una riprogettazione dell'intero sistema e delle due fasi e una nuova simulazione. Questo procedimento richiede un consumo di tempo e di soldi, inoltre molto spesso non risulta pratico. Così nasce il bisogno che la fase di simulazione off-line possa essere testata anche in assenza del prototipo e che si interfacce con le varie condizioni che il sistema può incontrare nel suo normale funzionamento. Infatti si parla di *simulazioni real-time* in tutti quei sistemi in cui diventa possibile sostituire i dispositivi fisici con dispositivi virtuali; questo riduce i costi di sostituzione e migliora la qualità dei sistemi fisici e di controllo, consentendo un più completo collaudo dell'intero sistema, e permette di fare "continuous testing", cioè senza interruzioni e in condizioni potenzialmente pericolose. Le simulazioni real-time consentono di far verifiche anche quando non si hanno prototipi e diventa una necessità se si vuole simulare sistemi realistici che rispondano all' ambiente circostante. Viene utilizzato molto spesso questo tipo di simulazione con Hardware-in-the-Loop (HIL).

1.1 Hardware-in-the-Loop (HIL)

Hardware-in-the-Loop è un metodo che esiste da non più di 15 o 20 anni e le sue radici si trovano nel settore del trasporto aereo. Col passare del tempo lo si è usato in molti alti settori (industria medica, settore industriale, settore gestione e trasformazione dell'energia, settore commerciale, industria aereonautica ed aereospaziale, automotive) e il motivo di questo ampio implemento è guidato da due fattori principali:

- i tempi di commercializzazione sempre più ridotti dovuti dall' esigenze di mercato;
- la continua crescita di complessità dei sistemi da realizzare.

Per Hardware-in-the-Loop (HIL) si intende tutte quelle tecniche di test delle unità di controllo elettroniche e meccaniche, che vengono collegate ad appositi banchi. Questi

banchi sono apparati complessi che riproducono in tutto o in parte, fisicamente e/o in software, il prodotto a cui sono destinate le unità per essere verificate.
Essi sono costituiti dai seguenti componenti:

- sensori, cioè i dispositivi di input;
- attuatori, cioè dispositivi che eseguono un'azione sul comando della centralina
- cablaggi elettrici ed elettronici e relative connessioni (cioè l'impianto elettrico e la rete che interconnette tra loro le centraline con le relative terminazioni dove collegare fisicamente sensori, attuatore e centealine);
- modelli, cioè programmi informatici che emulano tutto ciò che fisicamente non è implementato nel banco, "ingannando" i dispositivi effettivamente collegati.

Lo scopo delle prove HIL è di utilizzare i banchi per anticipare le verifiche sui componenti, sui sottosistemi e sui sistemi già nella fase di progettazione e prototipazione, senza attendere la disponibilità del prodotto finito. Infatti, i componenti reali installati rispondono ai segnali simulati come se stessero operando in un ambiente reale, poiché il simulatore Hardware-in-the-Loop non è in grado di distinguere segnali provenienti da un ambiente fisico da quelli prodotti dalla simulazione real-time.
In questo modo il metodo HIL consente di riprodurre sulla piattaforma le condizioni operative più diverse e di osservare il comportamento del sistema e dei singoli elementi, che molto spesso risulta difficile, o impossibile, fare solo col sistema reale.

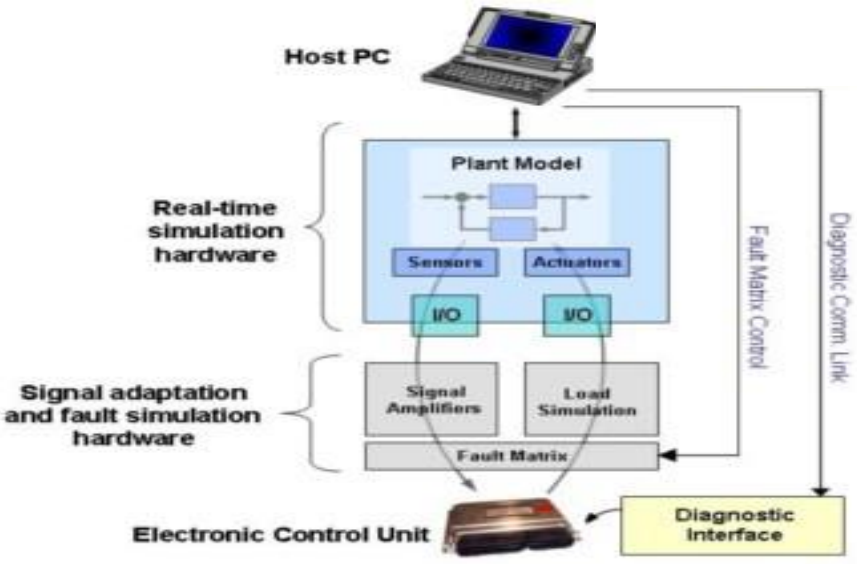


***Figura 1**-Tipico sistema di test HIL*

Uno dei vantaggi dell'approccio HIL è quello che permette un passaggio graduale dalla simulazione all'apparato reale in quanto consente di partire da una simulazione pura e durante la fasi di simulazione e di test consente di integrare gradualmente sottosistemi elettronici e meccanici reali nel ciclo, appena essi diventano disponibili.
Il processo HIL trova in commercio molte soluzioni di piattaforme programmabili, in base alle esigenze che il settore di impiego richiede, quindi risulta difficile avere un schema fisso nella sua realizzazione, ma si può lo stesso riassume il metodo Hardware-in-the-Loop nelle seguenti fasi:

- Il cuore di un simulazione HIL è rappresentato dalla componente di modellistica, ovvero rappresentazione matematica di tutti i sistemi dinamici relativi all'impianto;
- test del dispositivo sul processo simulato;
- implementazione dell'hardware sul processo reale (se la simulazione ha esito positivo).

Ciò che una simulazione HIL permette di fare è :

- la verifica dei sistemi prototipali;
- ottimizzazione delle strategie di controllo tramite analisi dinamica;
- verifica step by step delle singoli parti;
- possibilità di osservare grandezze nascoste nel sistema reale;
- validazione delle prestazioni del sistema di controllo;
- verifica dei limiti di stabilità e campo di lavoro;
- possibilità di effettuare test di vita accelerati del prodotto;

Bisogna ricordare però che il processo HIL è parte integrante del ciclo di progettazione e collaudo. La figura sotto rappresenta il ciclo di progettazione di un sistema tipico, ad esempio un sistema di controllo. Si vede che prima di passare al processo Hardware-in-the-Loop Testing un bravo ingegnere deve definire le specifiche del sistema da rispettare, validare in modo preliminare le strategie da usare e generare i codici del sistema e solo dopo potrà simulare con HIL e testare il sistema. Questo ciclo viene comunemente definito come "ciclo V".

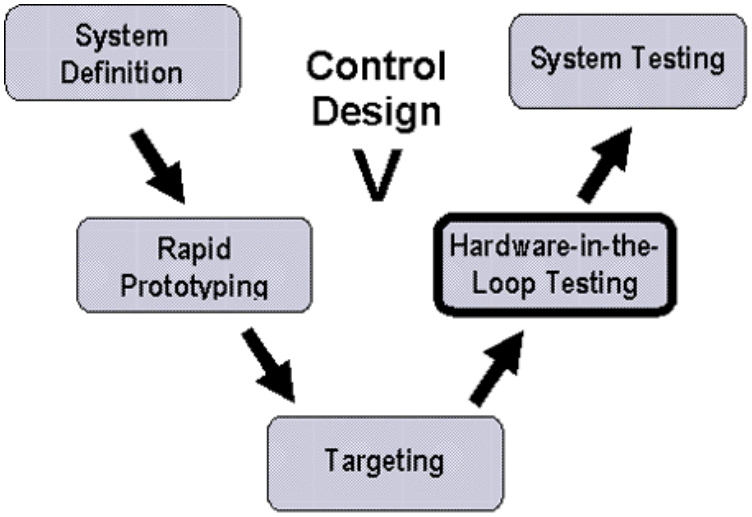


***Figura 2**-Ciclo progettazione standard*

In questi ultimi anni si è avuto un aumento della complessità dell'hardware dovuta da un passaggio da dispositivi analogici a quelli digitali. Infatti servono piattaforme più complesse ed efficaci per sviluppare e testare i sistemi real-time, spesso in stretto parallelo con lo sviluppo hardware.

1.1.1 Perché utilizzare tecniche di simulazione Hardware-In-The-Loop?

È un quesito determinante per capire la tecnologia real-time. In alcuni casi, il modo più comodo di sviluppare un sistema è quello di collegare il sistema stesso all'impianto reale. In molti casi è più efficiente la simulazione HIL. Il metro di giudizio per capire se è più efficacie sviluppare il sistema collegandolo direttamente all'impianto reale o sviluppandolo attraverso la simulazione HIL, sta nella formula che comprende i seguenti elementi:

- Costo
- Durata
- Sicurezza.

Se si va a pensare a tutti i settori dell'ingegneria coinvolti nello sviluppo, nella realizzazione e nell'implementazione di tali sistemi, dall'ingegneria gestionale a quelle di elettronica, di meccanica e di informatica, ci si rende conto che un errore nelle specifiche di progetto comporta il non raggiungimento degli obiettivi preposti e di

conseguenza perdite di denaro e di tempo in termini di attrezzature e di ore-uomo, oltre che a mettere a rischio l'impianto e l'incolumità delle persone.
Adottando il metodo Hardware-in-the-Loop si rende, quindi possibile superare le problematiche relative a :

- Programma di sviluppo;
- Complessità dell'impianto reale;
- Sviluppo iniziale delle risorse umane;

Per programma di sviluppo s'intende la maggior parte dei programmi per i quali non conviene attendere che il prototipo reale sia disponibile, per testare il sistema con il modello reale. Infatti, la maggior parte di nuovi programmi di sviluppo presuppone che la simulazione HIL avverrà in parallelo allo sviluppo dell'impianto. Ciò vuol dire che, nel momento in cui un nuovo prototipo sarà reso disponibile per il sistema di controllo da esaminare, il 95% della prova sarà già stata completata usando la simulazione Hardware-in-the-Loop.
In molti casi, il sistema reale è molto più costoso di un simulatore real-time altamente affidabile. Ovvero, risulta più economico effettuare le fasi di sviluppo e test collegati ad un simulatore HIL piuttosto che ad un impianto reale. Per esempio i fornitori di componenti elettronici, che lavorano nel ambito dell'elettronica di potenza, la simulazione HIL è una parte fondamentale dello sviluppo di questi componenti, permette di testarli in vari condizioni di utilizzo senza aver per forza la parte fisica dell' impianto reale.
La simulazione HIL è un punto chiave, quindi, nel processo di sviluppo delle risorse umane, un metodo per accertare il possibile impiego e la consistenza del sistema usando software ergonomici, risorse umane di ricerca, nonché di sviluppo, riferendosi in quest'ultimo caso all'operazione di raccolta dei dati utilizzati dall' Hardware-in-the-Loop per il test di componenti che avranno un'interfaccia umana.
Quindi, l'esigenza di stabilire una prima validazione del software di controllo senza ricorrere all'esecuzione di numerose e costose prove nell'ambiente di lavoro, ha portato allo sviluppo ed alla diffusione di sistemi di simulazione HIL, attraverso i quali è possibile testare componenti hardware, simulando il sistema da controllare con un modello capace di fornire in uscita dati plausibili e verosimili in sostituzione di quelli inviati dal sistema reale.

1.1.2 Hardware-in-the-Loop per elettronica di potenza

Come già detto in precedenza, una piattaforma HIL che permette di fare test in real-time, garantisce maggiore flessibilità in particolare durante la fase di test, e possiede una elevata sicurezza nei test. Questo in molti casi risulta essenziale, quasi indispensabile, per la fase di sviluppo e test dei sistemi. Uno degli ambiti per cui lo è diventato, è l'elettronica di potenza. Per elettronica di potenza si intende sia una branca dell'elettronica e allo stesso tempo una tecnologia associata alla conversione, al controllo e alla modifica efficiente della potenza elettrica disponibile in ingresso a un sistema fisico, circuito elettrico o elettronico, per trasformarla in un'uscita desiderata.

I sistemi di conversione e di controllo di potenza si trovano in molti settori e ovunque ci sia la necessità di modificare forma dell'energia elettrica e di controllarla. Sono nate diverse soluzioni di piattaforme HIL cercando di gestire e soddisfare le esigenze del settore in cui vengono utilizzate.

In questi ultimi anni le rete di trasmissione e gestione dell' energia elettrica tradizionali hanno subito dei cambiamenti dovuto da alcuni fattori:

- una riduzione degli idrocarburi (decarbonizzazione) nella generazione dell'energia elettrica;
- una crescente domanda di energia elettrica nelle economie in via di sviluppo;
- una elettrificazione del settore di trasporto.

Così i sistemi tradizionali che erano poco centralizzati e controllati hanno lasciato posto a sistemi più moderni che hanno portato un aumento della complessità della distribuzione dell'energia e delle reti di trasmissione, anche per mezzo di un maggior uso dell'elettronica di potenza come nell' HVDC.

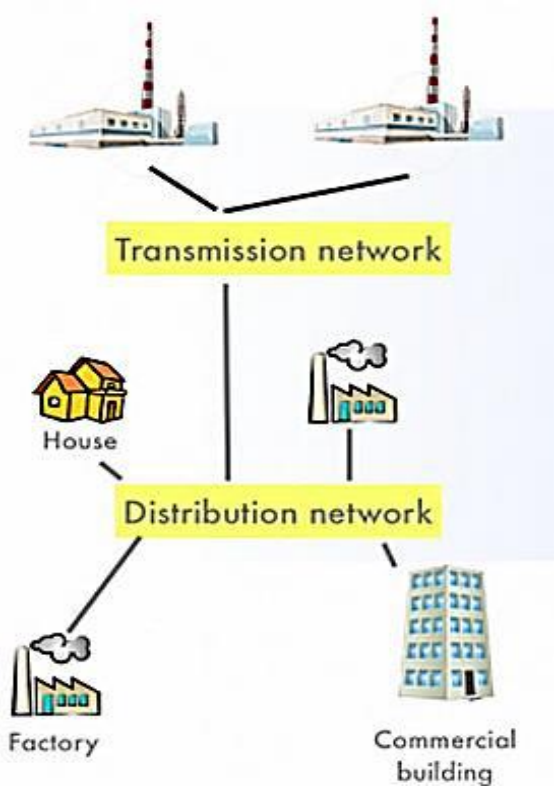


***Figura 3**-Schema rete elettrica tradizionale*

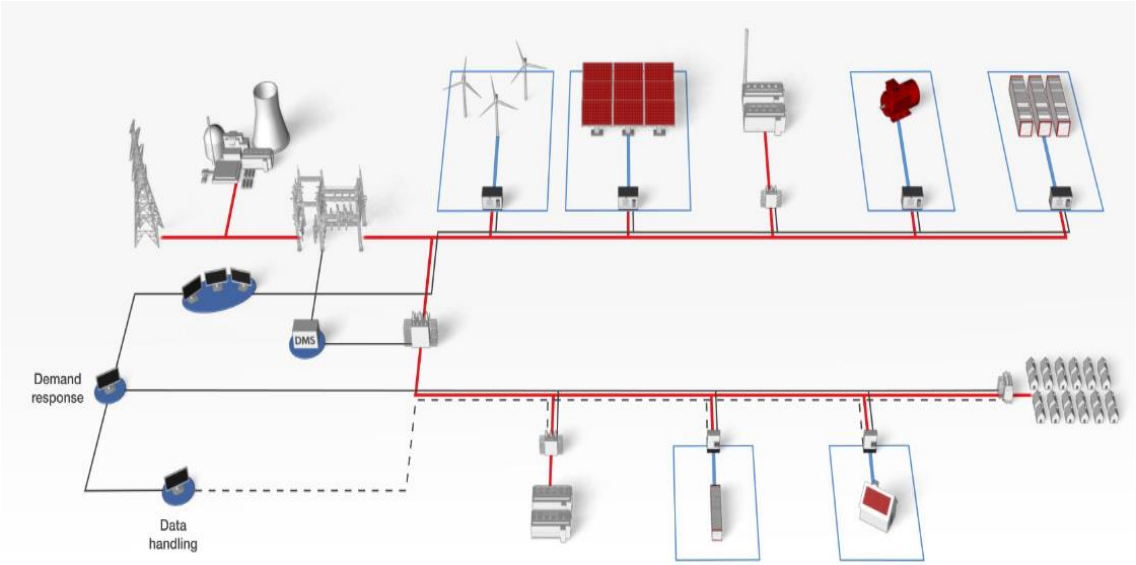


***Figura 4**-Schema rete elettrica moderna*

I simulatori HIL per le reti di trasmissione e gestione dell'energia elettrica sono stati utilizzati per decenni per testare nuove progettazioni dei sistemi di rete, soprattutto per sistemi di trasmissione ad alta tensione, e per testare nuovi dispositivi elettronici di potenza destinati a stabilizzare i sistemi di alimentazione e per interconnettere i sistemi

che utilizzano HVDC. Prima del 1990, la simulazione HIL era tradizionalmente effettuata utilizzando simulatori analogici, che erano piuttosto costosi e difficili da utilizzare. Negli anni novanta da diversi centri di ricerca sono stati sviluppati simulatori in real-time digitali per diminuire il costo dei simulatori del sistema di alimentazione e per facilitare la loro manutenzione e il funzionamento.

I sistemi HVDC, acronimo per High Voltage Direct Current, sono sistemi di trasmissione di energia elettrica in corrente continua, anziché in corrente alternata. L'impiego classico dei sistemi HVDC è nella trasmissione di grandi potenze su lunghe distanze dato che i costi complessivi del sistema di trasmissione e le relative perdite in linea sono minori rispetto ad un sistema in corrente alternata. Il principale vantaggio della connessione in HVDC è rappresentato dal fatto che non c'è nessun limite di stabilità legato alla quantità di potenza trasmessa ed alla lunghezza della linea di trasmissione. Questi sistemi trasformano la corrente proveniente dai classici sistemi di generazione dell'energia elettrica come centrale idroelettriche, nucleare e ultimamente molto più spesso da fonte di energia rinnovabile come dall'eolico, solare ecc.; per poi trasformala per le utenze finale in corrente alternata. I dispositivi HVDC, che permettono questo, si basano sulla conversione di corrente alternata in corrente continua e viceversa tramite convertitori AC/DC e viceversa che includono tiristori di potenza, che sono il cuore di una stazione di conversione, e di filtri DC ed AC.

***Figura 5**-Stazione HVDC*

Per testare e sviluppare i dispositivi HVDC solitamente si fa uso di una piattaforma HIL, perché garantisce all'utente di effettuare tutti i passi necessari per preparare ed eseguire la simulazione, e di analizzare i risultati della simulazione, il tutto senza l'utilizzo di potenza. Infatti il simulatore va rappresentare il sistema di alimentazione dove viene poi interfacciato con schede hardware precedentemente programmate dal software che vanno a sostituire i componenti reale dei controlli e di conversione del sistema.

Una piattaforma molto diffusa è quella proposta dalla RTDS, azienda che da oltre trent'anni si occupa della gestione di sistemi HVDC in tutto il mondo. Propone un simulatore, RTDS simulator, che è stato originalmente sviluppato per modellare sistemi HVDC. Nel corso degli ultimi due decenni, il simulatore ha rivoluzionato il processo di test ad anello chiuso di sistemi di relè di protezione e di controllo, e funziona in real-time, fornendo risultati accurati su una gamma di frequenza da DC ai chilohertz. La forma della piattaforma è quella di un "scafale", nel quale al suo interno può contenere schede di input / output per essere interfacciato con apparecchiature esterne come i controller, relè di protezione e/o amplificatori, inoltre contiene i componenti di entrata di potenza, alimentatori e altri componenti necessari.

Figura 6-*RTDS Simulator*

Il simulatore utilizza unicamente il software RSCAD, che va a comprendere diversi moduli che permettono tutti gli aspetti della simulazione real-time per essere creati,

eseguiti, controllati e analizzati senza l'uso di prodotti a terzi. È uno strumento HIL ideale per la simulazione e la sperimentazione in real-time di dispositivi HVDC. Attualmente RSCAD è in grado di accettare e convertire i dati da altri programmi come PSS, MatLab-Simulink.

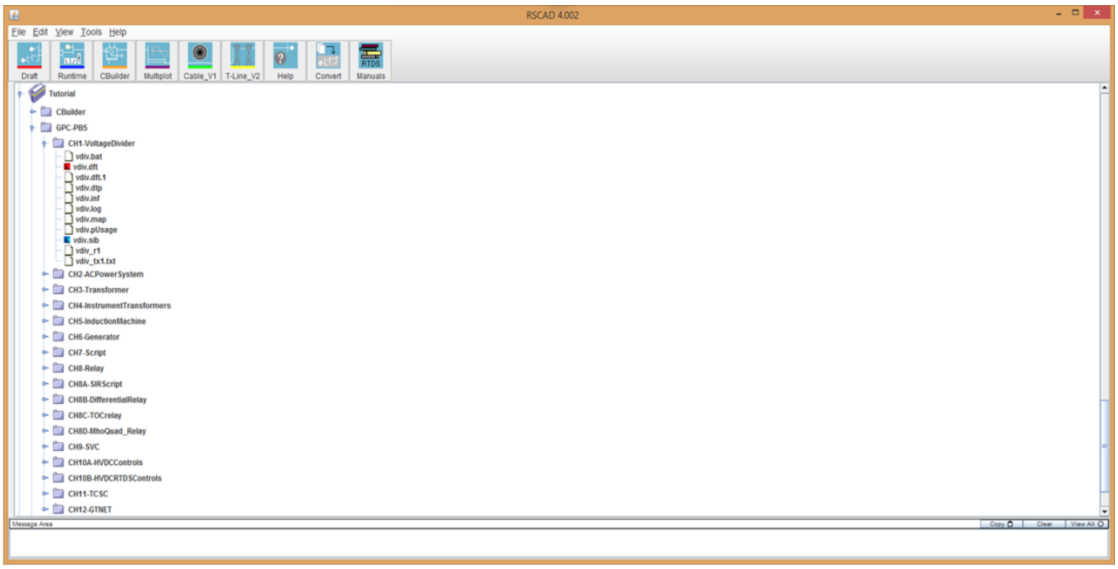


Figura 7-*Ambiente di sviluppo RSCAD*

Altre soluzioni HIL valide sono presenti in commercio per poter testare al meglio i sistemi di trasmissione moderni dell'energia elettrica dovuto all'interoperabilità delle fonti di energia rinnovabile (solare, eolica, ecc) e per monitorare, proteggere e controllare maggiormente i dispositivi ed i sistemi elettrici ed elettronici di potenza.
Una prima soluzione è quello che propone la Typhoon-HIL, un' azienda nata solo nel 2008 ma che si occupa di sistemi HIL per elettronica, micro reti e reti di distribuzione dell'energia. La piattaforma HIL è Microgrid Testbed, che ha come scopo principale quello di testare in modo completo la componente hardware, software e firmware del sistema simulato in tutte le condizioni operative ed individuare difetti sia nella modalità isolata che quella connessa alla rete.

Figura 8-*Microgrid Testbed*

È un insieme di dispositivi HIL con relè di protezione integrato, controllori di rete e micro rete e convertitori per i sistemi di energia solare , gruppi elettrogeni, per batterie e celle a combustione

Microgrid Testbed è in grado di eseguire tutti i test e di generare i rapporti di prova anche in modalità completamente automatica, aumentando così la produttività e migliorando ulteriormente la copertura dei test.

Per la simulazione, la Typhoon HIL fornisce un unico ambiente di sviluppo a livello software completamente integrato con l'hardware , il Typhoon HIL Control Central. Uno strumento che permette alte prestazioni e una stabilità numerica a lungo temine, possiede sia un Centro di Controllo dove ha luogo tutte la modellazione e la sperimentazione, sia un compilatore incredibilmente veloce che converte i modelli in processi leggibile al hardware.

Typhoon HIL Control Central consente di rilevare già nella modellistica del sistema i disturbi di rete, cortocircuiti ecc., poiché possiede una ricca libreria di funzione matematiche che permettono di quantificare le prestazioni del sistema in un ampio spettro in condizioni di esercizio, dalle condizioni operative standard, così come in condizioni di guasto (interno ed esterno).

Figura 93-*Typhoon Control Central*

Dispone al suo interno di una TestSuite Toolbox che permette di pre-certificare il sistema di controllo dell'inverter, questo permette di sapere quando si è pronti per testare il convertitore presso un laboratorio accreditato. La pre-certificazione è verificata dall' Istituto austriaco AIT of Technology GmbH.

Un'altra soluzione viene proposta dalla OPAL-RT,un azienda che da oltre 15 anni si occupa di sviluppare simulatori in real-time, di fare test in Hardware-in-the-Loop per apparecchiature elettriche, elettromeccaniche e sistemi elettronici di potenza.

L'obiettivo del simulatore è quello di verificare e certificare la funzionalità, le prestazioni, la qualità e la sicurezza del programma di controllo e il sistema di protezione del software. A tal fine, il controllo effettivo e di protezione sotto test è collegato ad un simulatore attraverso interfacce di corrente e tensione nelle stesso modo come nel sistema reale. Il simulatore simula con alta precisione e fedeltà lo stato stazionario e il comportamento in transitorio del sistema modellato in condizioni normali e difettosi. Ricreando l'ambiente realistico, il controller è "ingannato" a credere che è collegato al sistema fisico reale.

Il simulatore HIL è sufficientemente flessibile per essere utilizzato per una vasta gamma di applicazioni di micro reti, componenti dell'elettronica di potenza e dettagliati studi di stabilità alla rete. Questa soluzione può quindi servire per diversi progetti senza la necessità di investimenti in funzione o apparecchiature aggiuntive. La piattaforma di simulazione proposta è eMEGAsim, è un simulatore scalabile digitale in tempo reale che integra un super computer con un multi-processore real-time, librerie di modelli,

risolutori ad alta velocità, e una vasta gamma di moduli di I / O, in grado di soddisfare i severi requisiti di sistemi di grande potere, i sistemi di energia rinnovabile, e la simulazione di convertitori HIL. eMEGAsim è ideale per i sistemi di alimentazione a corrente alternata e sistemi di alimentazione a corrente continua, e sistemi di alimentazione industriali complessi.

eMEGAsim consente agli ingegneri di progettare in modo efficiente i sistemi di alimentazione di prova, strumenti di monitoraggio, sistemi elettronici, relè controller e dispositivi di protezione in modo efficiente, riducendo i costi.

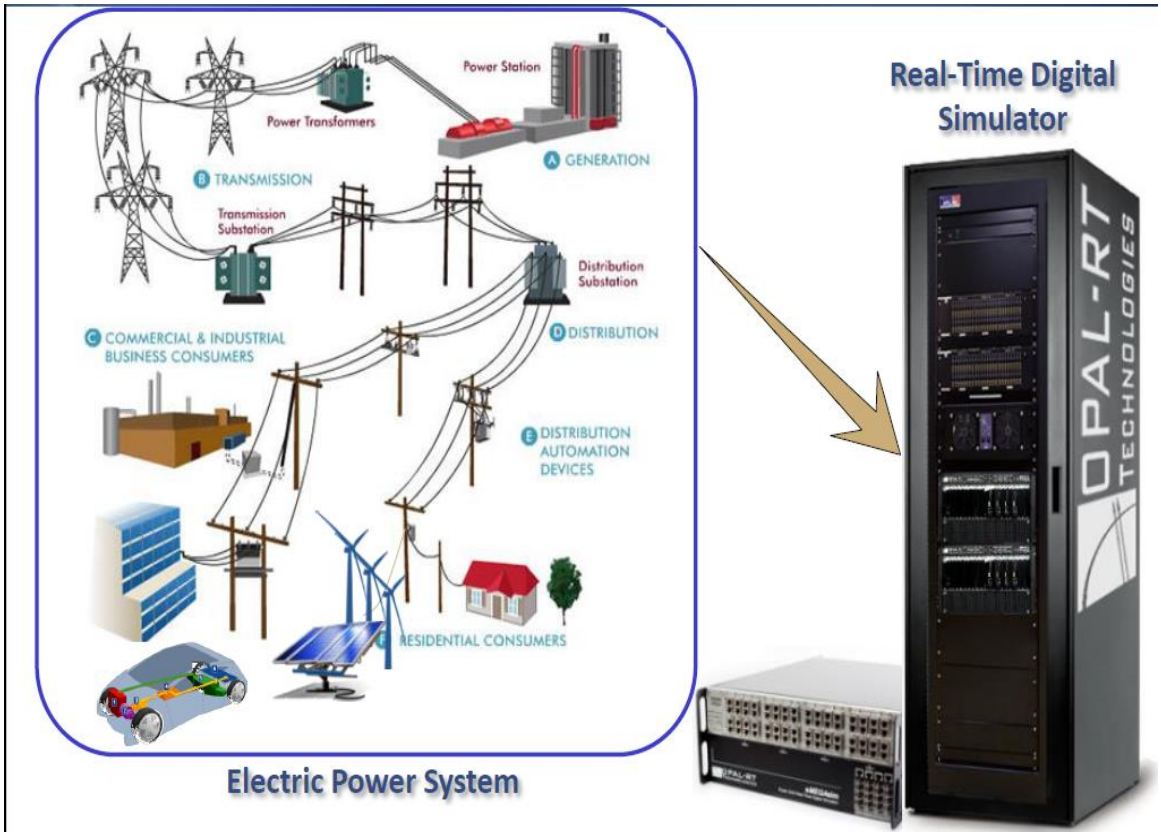


***Figura 4**-Simulatore dell' OPAL-RT*

Il simulatore ha una ricca biblioteca di modelli elettronici di potenza, in cui riesce a simulare transistor elettromagnetici ed elettromeccanici di rete elettriche a corrente alternata e a corrente continua; simula sistemi HVDC e micro reti. Al suo interno incorpora librerie di blocchi elettrici per simulare le macchine industriali, le unità le linee, trasformatori, motori, carico, o qualsiasi elemento che forma sistemi di alimentazione, componenti elettronici di potenza, sistemi di controllo e dispositivi di protezione e relè. Fornisce un'interfaccia con SimPowerSystems, PLECS, Simscape, e EMTP-RV. Questo permette di sfruttare modelli e risolutori disponibili in strumenti di

simulazione standard del settore. Inoltre fornisce un'interfaccia con MATLAB e Simulink per lo sviluppo, prototipazione e test di dispositivi di protezione, relè e sistemi di controllo in anello chiuso.

L'ultima soluzione di piattaforma HIL che si vorrebbe proporre, rimanendo sempre per il settore delle trasmissioni dell'energia elettrica, è quella della National Instruments, azienda che da oltre 40 anni si occupa di cercar soluzioni in tutti i campi ingegneristici, anche nelle simulazioni in real-time.

La piattaforma HIL è la CompactRIO, si tratta di un strumento che ha come cuore pulsante il controller costituito da un processore real-time per elaborazione del segnale e da un FPGA che si interfaccia con i moduli I/O.

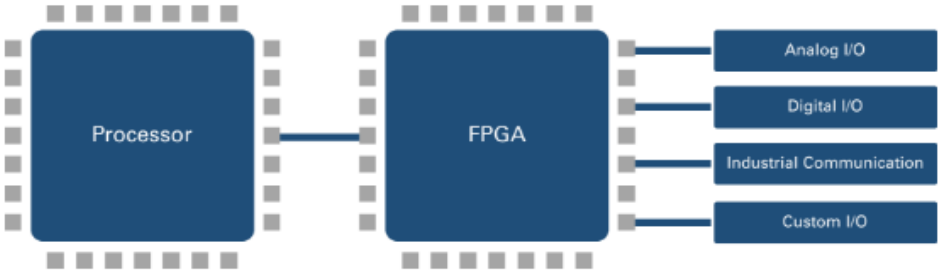


Figura 11-*Schema controller CompactRIO*

Il controller fornisce funzionalità di controllo e di acquisizione di dispositivi elettrici ed elettronici in una soluzione compatta. Elimina la necessità si sottosistemi separati collegando i componenti direttamente ad esso.

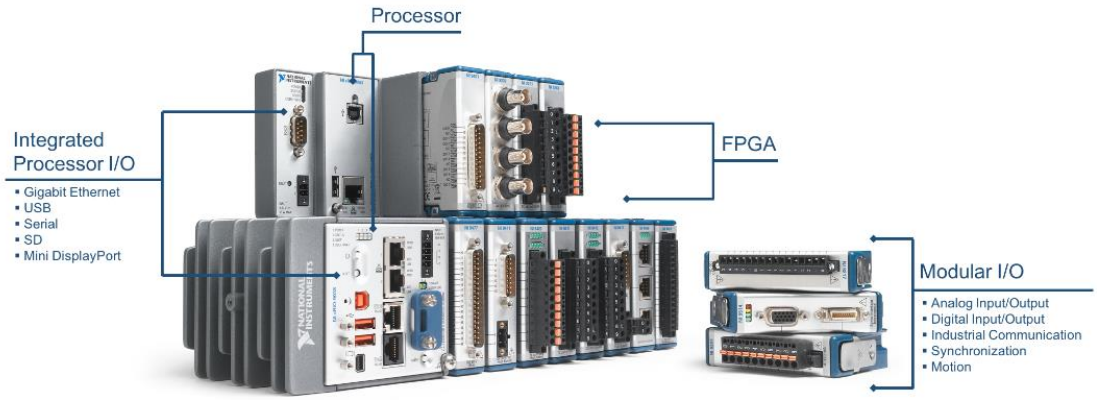


Figura 12-*CompactRIO*

Il software di progettazione è LabVIEW, che consente di superare le tradizionali architetture eterogene programmando il processore real-time e FPGA integrato, che interagire con I/O. LabVIEW è utile per la progettazione di efficienti prototipi ed al suo interno ha un'ampia libreria di componenti dell'elettronica di potenza. Riesce ad approcciarsi ad una varietà di programmi tra cui sviluppo grafico, file script .m e l'integrazione di C / C ++ e Python e il codice HDL.
Tutte le soluzioni HIL proposte nel settore delle reti elettriche per elettronica di potenza mostrano alcuni aspetti in comune:

- L'obiettivo di una completa progettazione, messa a punto e sviluppo dei componenti (controllo, protezione, servizi di energia) del sistema da simulare;
- Test di controllo, sistema di monitoraggio, protezione hardware in anello chiuso con il simulatore;
- Testare una varietà di scenari. Difficile da poter fare su una rete elettrica reale: guasti, operazione isolate, load rejection;
- Aver un'ottima interazione tra l'ambiente di progettazione e la componete hardware.

Sono state presentate e descritte alcune delle soluzioni Hardware-in-the-Loop più utilizzate per la gestione e trasformazione delle reti elettriche. Di seguito verrà riportato un caso di studio di come queste piattaforme possono essere implementate. Il caso di studio tratta della simulazione HIL di una rete per un inverter fotovoltaico utilizzando RTDS simulator. La Simulazione HIL permette di testare l'inverter fotovoltaico in un ambiente controllato prima di essere collegato al sistema fisico reale. Un modello software del sistema reale, in cui il dispositivo fisico opererà, si sviluppa sul RTDS simulator. Bisogna fare molta attenzione al metodo utilizzato per lo scambio dei segnali di tensione e di corrente nell'interfaccia HIL, alle potenze del dispositivo di prova e all'amplificatore scelto poiché sono punti critici nella simulazione.
Il sistema è costituito da un inverter fotovoltaico (255W) collegato alla rete, un convertitore digitale-analogico incluso nella parte del simulatore, che forniscono segnali analogici scalati verso il basso per livelli elettronici entro 10 Vpk. Questi segnali vengono forniti in ingresso ad un amplificatore di tensione lineare 1kVA che fornisce la tensione richiesta, la corrente e i livelli di frequenza dell'inverter. C'è la presenza di sensori che misurano la tensione o corrente ottenuti dall'inverter, che vengono poi fatti

passare attraverso un convertitore analogico-digitale (nel simulatore) per chiudere il circuito. I filtri implementati in hardware/software sono necessari per eliminare il rumore nell'interfaccia HIL.

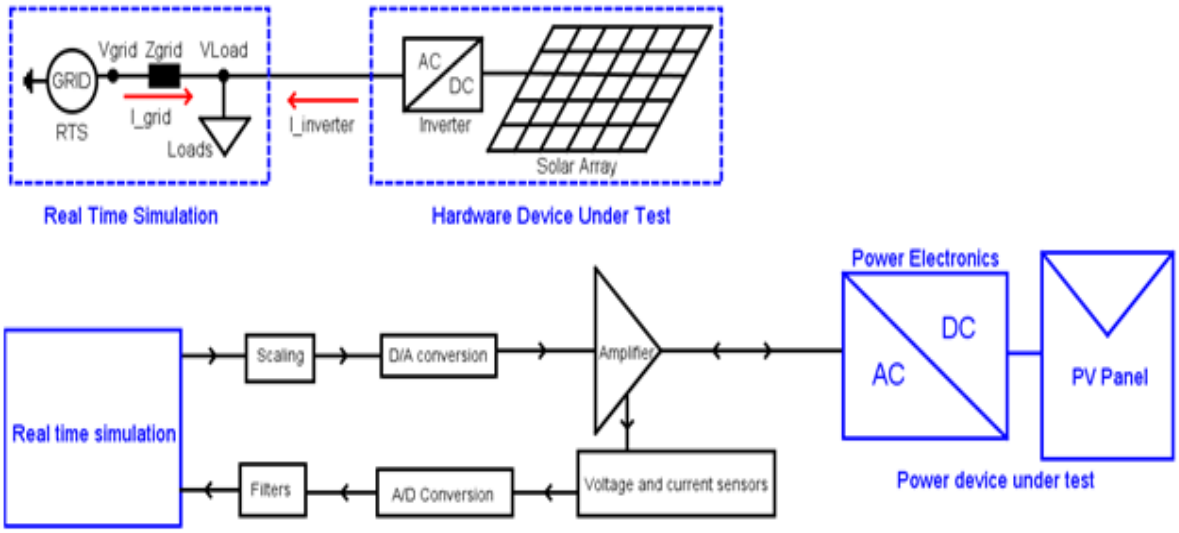


Figura 13-*Interfaccia HIL tra un RTDS e inverter fotovoltaico*

Il lato DC dell'inverter è collegato ad un panello solare con una fonte di luce ad alogenuri metallici, usato come un simulatore solare; mentre il lato CA dell'inverter è collegato al terminale di uscita dell'amplificatore che fornisce la simulazione di rete 240Vrms, 60 Hz. La corrente dell'inverter misurata dall'amplificatore viene inviata alla rete simulata nel RTDS simulator.

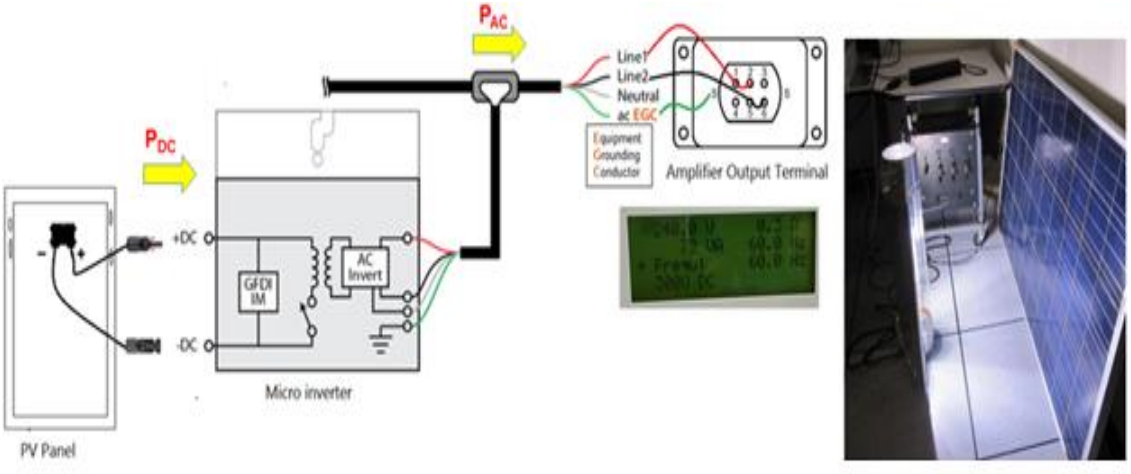


Figura 14-*Messa a punto di un test di una simulazione HIL*

La *Figura 15* mostra la corrente dell'inverter osservato nel simulatore sovrapposto col rumore dall'interfaccia HIL. Un filtro passa basso è utilizzato per eliminare il rumore sul

segnale di corrente. Il filtro migliora la stabilità della simulazione HIL ma attenua l'ampiezza del segnale di corrente e aggiunge un ritardo aggiuntivo. L'ordine di grandezza dell'attenuazione e tempo di ritardo introdotto dal filtro è determinato dalla frequenza di taglio selezionata. La *Figura 16* mostra una differenza del 50% della potenza dell'inverter calcolata nel RTDS simulatore quando la frequenza di taglio del filtro passa da 500 Hz a 500kHz. Frequenze di taglio più basse si traducono in elevati ritardi di tempo e una maggiore attenuazione, che va ad influenzare sulla precisione della simulazione HIL.

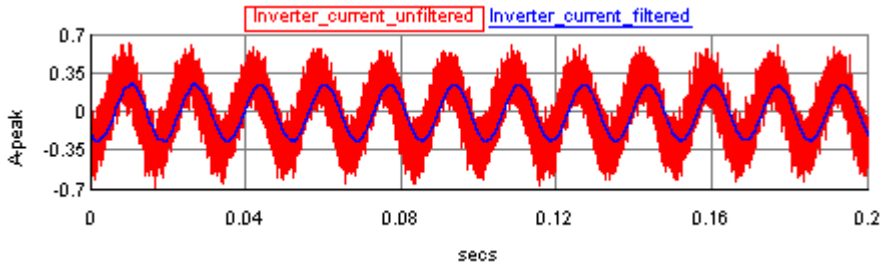


***Figura 15**-Corrente dell'inverter misurata nel simulatore*

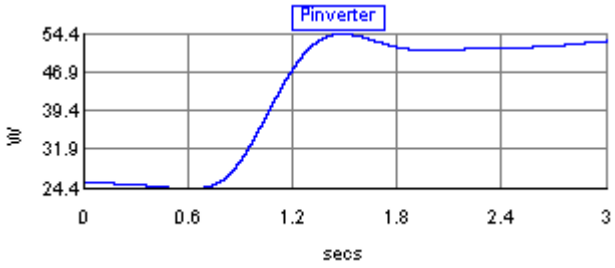


***Figura 16**-Potenza inverte osservata dal simulatore con l'incremento del 50%*

Inoltre la simulazione permette di sperimentare il dispositivo con diversi tipologie circuitali, ad esempio la *Figura 17* mostra la risposta del convertitore a una linea di guasto della messa a terra. La risposta attuale dimostra che l'inverter rimane collegato alla rete quando la durata del guasto è di cinque cicli. Il punto in cui si scollega dalla rete può essere visto quando la durata del guasto è stata aumentata a 60 cicli. Va notato che gli scenari di emergenza da testare nell'interfaccia HIL dovrebbe essere entro i limiti di protezione del dispositivo in prova e le valutazioni di amplificazione per evirare un funzionamento instabile.

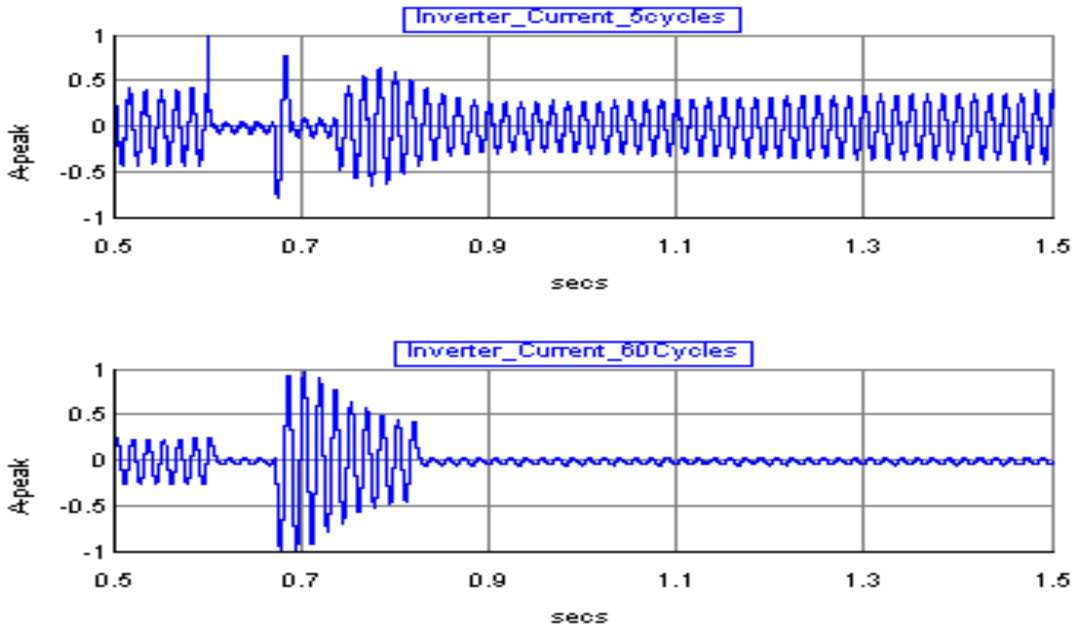


Figura 17-*Risposta dell'inverte della parte AC di 5 e di 60 cicli, con linea di messa a terra in condizioni di guasto*

1.2 Modelli continui, discreti ed ibridi

La componente di modellistica diventa parte importante nelle realizzazione delle simulazione real-time, ovvero si parte dalle rappresentazioni matematiche dei sistemi per poter andar a costruire dei modelli utili per la realizzazione dei sistemi HIL.

Convenzionalmente la realizzazione di un modello di un sistema, per poter poi essere utilizzato in una simulazione real-time, viene rappresentato attraverso la scelta di uno dei due approcci classici: modello continuo e modello discreto. Anche se un terzo modello, quello ibrido, sta iniziando ad essere utilizzato molto più spesso nella fase di simulazione.

Tutti i tipi di modello richiedono che il sistema di simulazione sia caratterizzato da uno stato di sistema che cambi con l'avanzare del tempo. È la natura di questo cambiamento che distingue i modelli.

Nel modello discreto, si assume che lo stato del sistema si modifichi ad un determinato istante di tempo e che rimanga invariato tra un istante di tempo e l'altro. Questo cambiamento nello stato del sistema avviene istantaneamente, come se fosse il risultato di un evento che lo attiva. La gestione del tempo in una simulazione discreta si basa sulla conoscenza dei tempi di eventi futuri, che sono noti a priori o mantenuti in una "coda di eventi prioritari". I primi linguaggi di simulazione basati su un processo discreto sono stati realizzati negli anni 60, hanno preso piede soprattutto per la simulazione dell' elettronica digitale, del controllo digitale, e dei sistemi a dati

campionati che possono essere realizzati utilizzando software di progettazione digitale come VHDL, Verilog, oppure System Generator, che combinano le specifiche del sistema con un simulatore.

Nel modello continuo lo stato del sistema permuta in modo continuo, come definito dalle equazioni differenziali del modello. In realtà si può parlare di sistemi “quasi” continui poiché si cerca di adattare il passo temporale discreto ad algoritmi di integrazione numerica che cercano di generare soluzioni approssimate alle equazioni differenziali. Le simulazioni continue si basano su soluzioni di equazioni differenziali ordinarie o parziali per i quali il tempo è un evoluzione di una variabile continua senza variazioni istantanee da un valore del tempo a quello successivo, al contrario del comportamento di un modello discreto in cui il tempo salta da un evento al successivo. Esistono molti algoritmi disponibili per risolvere le equazioni differenziali, e la scelta di algoritmo è una delle decisioni chiave effettuate dall'utente. Una delle scelte affrontate dagli utenti, che usano l’approccio continuo, è se utilizzare un algoritmo che avanzi nel tempo con incrementi uguali (un algoritmo a passo fisso) o uno che vari la dimensione di incrementi di tempo per soddisfare le tolleranze fornite dall'utente (un algoritmo a passo variabile). Le routine a *passo variabile* moderne utilizzano spesso un approccio in cui si va a vedere l’errore di stima dato dalla differenza della soluzione di ordine “n” stimata con la soluzione di ordine “n-1” calcolata, cercando di aggiustare il più possibile il passo. La giustificazione per questo è che questa differenza approssima il valore dell'ultimo termine della serie di Taylor prima del troncamento. Ciò viene considerato una stima conservativa del troncamento dell'errore totale. Per esempio, la routine predefinita sia in MATLAB che in Simulink è *ode45*, un metodo di questo tipo che confronta approssimazioni del quinto e quarto ordine alle soluzione. Invece le routine a *passo fisso* sono più semplici e sono favorite per la simulazione in tempo reale, perché risultano in un tempo di calcolo costante, anche se le routine passo variabile sono più sicure, e spesso usate come default in molti sistemi di simulazione. I metodi discussi finora sono autonomi all'interno di ogni passo e sono chiamati i metodi di un *singolo passo*. Per ogni passo temporale il simulatore esegue una serie di comandi:

1. Legge input e/o genera output;
2. Risolve il modello di equazioni;
3. Aspetta l’inizio di un nuovo passo;

Una spiegazione semplificata di questa routine suggerisce che in ogni passo temporale uno stato del modello venga campionato e risolto. Se non sono soddisfatte tutte le condizioni temporali di simulazione in tempo reale, si possono verificare discrepanze tra i risultati del simulatore e le risposte della sua controparte fisica. Alcuni metodi di calcolo si distribuiscono su più di un passo e sono chiamati metodi multi-step. Ad esempio, il valore dello stato del sistema al passaggio successivo può essere determinato in termini di stato del sistema ai tempi di passo corrente e precedente. Alcuni algoritmi richiedono dati degli ultimi tre o quattro passi. Questi metodi sono spesso convenienti (in termini di quantità di calcolo necessaria per raggiungere un determinato accuratezza), ma richiedono procedure speciali per avviarli poiché i dati passati non sono disponibili all'inizio della simulazione. Questi metodi possono anche causare problemi con simulazioni ibride e real-time.
Un modello ibrido è un modello che contiene caratteristiche di entrambi i modelli discreti e continui.

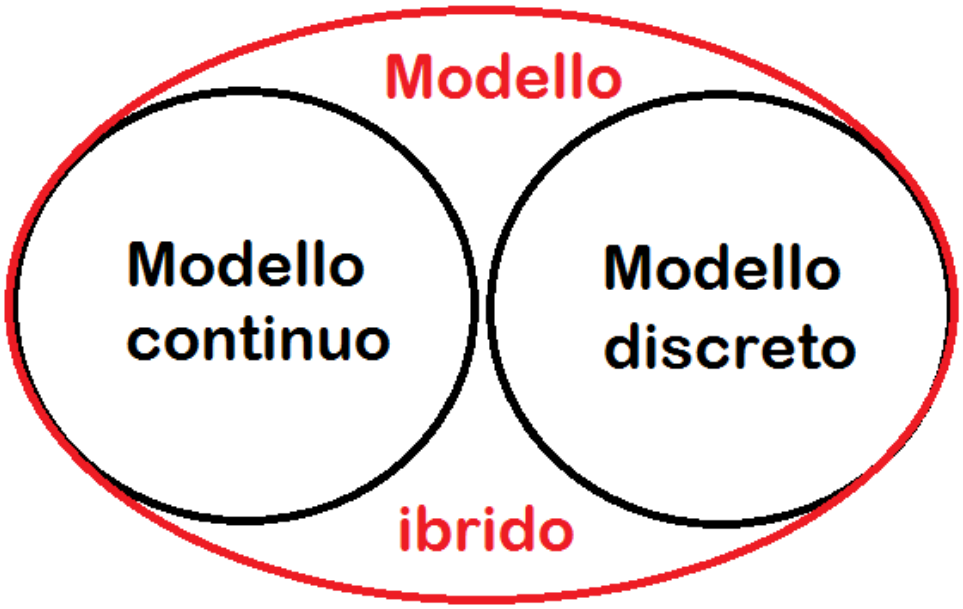


Figura 18-*Modelli di simulazione*

Ai fini delle simulazioni, un sistema ibrido può essere visualizzato in alcuni casi come un modello sostanzialmente discreto contenente uno o più stati che cambiano in modo continuo tra gli eventi, e/o in altri casi come un modello continuo in cui gli stati cambiano continuamente per la maggior parte del tempo, ma in cui uno o più stati

possono cambiare istantaneamente. In altri casi ancora, i processi discreti e continui possono essere equilibrati in un modello integrato che è uniformemente distribuito tra componenti discreti e continui. Molti prodotti software di simulazione attualmente disponibili forniscono un supporto completo sia per modelli discreti che continui con caratteristiche aggiuntive limitate che supportano l'approccio alternativo. L'equilibrio tra elementi discreti e continui nel modello viene spesso dettato dalla scelta del software per eseguire la simulazione. Xilinx System Generator insieme a Simulink consentono una buona collaborazione, grazie anche a un'interfaccia Matlab intuitiva al lavoro di simulazioni ibride.

1.3 Una metodologia alternativa al HIL: VHDL

In concomitanza con lo sviluppo della metodologia Hardware-in-the-Loop, a causa dell'esigenza di progettare e testare circuiti integrati sempre più complessi, si sono sviluppate in alternativa dei metodi di progettazione standard per i circuiti digitali. In particolare si è sviluppato il linguaggio VHDL. Il VHDL è un linguaggio per la descrizione dell'hardware che può essere utilizzato per la documentazione, la simulazione e la sintesi di sistemi digitali. Inizialmente, nei primi anni '80, lo sviluppo del VHDL è stato supportato dal dipartimento della difesa statunitense, nell'ambito di un progetto denominato VHSIC (Very High Speed Integrated Circuits). VHDL è infatti un acronimo che sta per VHSIC Hardware Description Language. Nel 1987 il VHDL è stato adottato come standard della IEEE.

Il VHDL è stato introdotto come linguaggio standard per la documentazione di sistemi digitali complessi. Il linguaggio è nato quindi con lo scopo di fornire una descrizione non ambigua di un sistema digitale, che potesse essere interpretata univocamente dai vari progettisti impiegati nello sviluppo del sistema stesso. Una delle caratteristiche richieste al VHDL è la possibilità di simulare il sistema descritto, sia a livello funzionale sia portando in conto i ritardi del circuito. Negli anni seguenti, oltre che per la documentazione e la simulazione, il VHDL ha assunto un ruolo sempre più importante nella fase di sintesi dei sistemi digitali. Un programma di sintesi consente, a partire da una descrizione comportamentale del sistema, di ottenere automaticamente una descrizione del circuito a basso livello mediante una netlist in cui il sistema viene descritto come interconnessione di porte logiche elementari appartenenti ad un'opportuna libreria. Il ciclo di sviluppo di un sistema integrato diviene simile a quello di un programma software in cui si parte da una descrizione in un linguaggio di

programmazione (C, PASCAL,…) per ottenere, dopo una fase di compilazione, una descrizione in linguaggio macchina.

L'utilizzo di sofisticati programmi di sintesi è stato uno degli strumenti più importanti che ha consentito lo sviluppo di circuiti integrati sempre più complessi, consentendo al progettista di concentrarsi sulla descrizione ad alto livello del sistema, esplorando come le diverse scelte architetturali possano influire sulle prestazioni del circuito, disinteressandosi dei dettagli implementativi. Il VHDL consente infatti di descrivere efficacemente sistemi complessi a cui corrispondono netlist di centinaia di migliaia o milioni di porte logiche elementari, così come in un programma software ad alto livello è possibile ottenere facilmente programmi in linguaggio macchina costituiti da milioni di istruzioni elementari a partire da un listato di poche centinaia di righe. Un vantaggio di questo linguaggio è legato all'utilizzo di programmi di sintesi, infatti la descrizione di un sistema digitale in VHDL può essere indipendente dalla particolare tecnologia prescelta per l'implementazione del circuito, così come un programma descritto in un linguaggio ad alto livello può essere complito ed eseguito su piattaforme hardware differenti. Ciò è possibile poiché è un linguaggio standard.

Un ulteriore vantaggio è quello di un miglioramento nella qualità del progetto, poiché consente sia di analizzare varie alternative, sia concede più livelli di astrazione e permette un riutilizzo e una condivisione dei blocchi già sviluppati.

Anch'esso come linguaggio presenta dei limiti. Infatti è un linguaggio poco "immediato" e sinteticamente pesante. Un altro limite è dovuto dal fatto che essendo uno "Standard Aperto" si possono definire strutture e/o tipologie di variabili non direttamente collegabili alla struttura del circuito.

Da non dimenticare che il VHDL è nato come linguaggio per la documentazione dei sistemi digitali e solo dopo in un secondo momento sono stati introdotti i programmi di sintesi, non deve stupire il fatto che non tutti i costrutti del VHDL siano sintetizzabili.

Capitolo 2

Un ambiente di sviluppo per i modelli HIL

Dopo aver parlato nel capito precedente della simulazione real-time e di come essa viene lungamente utilizzata nella simulazione HIL, e di aver fatto un accenno ai tipi di modelli di simulazione real-time, in questo capitolo, invece si prenderà in considerazione un ambiente di sviluppo che permette di progettare la modellistica per la simulazione real-time ibrida, Xilinx System Generator. Esso interagisce con l'ambiente di MATLAB e permette una buona integrazione con le librerie di Simulink consentendo di lavorare un segnale continuo in un segnale discreto e digitalizzato. Si realizzerà degli esempi di Xilinx System Generator con lo scopo di far vedere l'utilizzo di questo software nella realizzazione della componete di modellistica delle simulazione real-time.

2.1 MATLAB-Simulink

MATLAB (MATrix LABoratory) è un linguaggio di programmazione per applicazioni scientifiche (elaborazione numerica dei segnali, progetto di simulatori, sintesi di sistemi di controllo, ecc.). MATLAB è un interprete di comandi, i quali possono essere forniti interattivamente o contenuti in files su disco (M-files). Comprende un vasto set di funzioni predefinite e numerose librerie (toolbox) per svariate applicazioni. Le potenzialità di MATLAB possono essere facilmente estese (è semplice creare nuovi toolbox) ed è possibile convertire un programma MATLAB in codice C e C++ in modo automatico. Di seguito riportato la finestra di MATLAB che si andrà ad utilizzare.

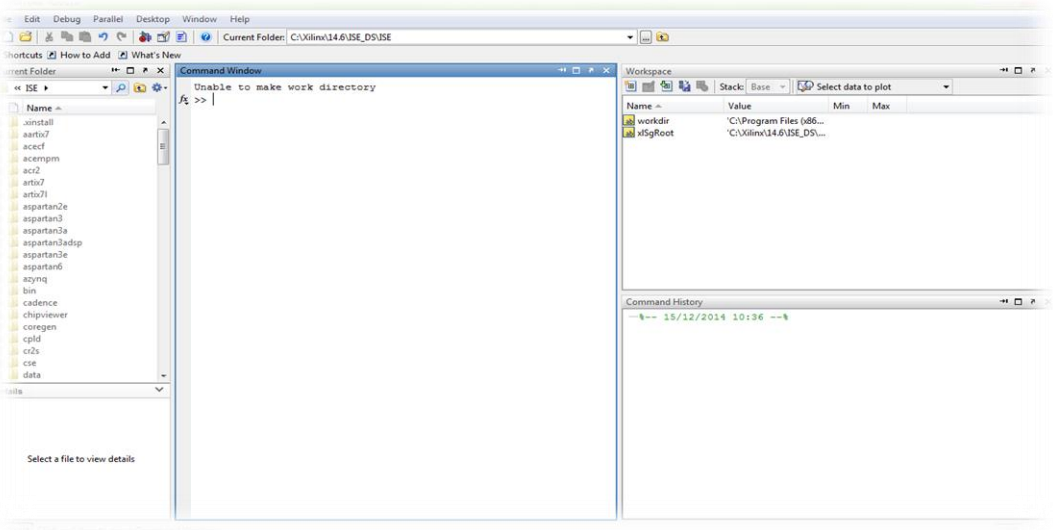


Figura 19-*Finestra apertura di Matlab*

Simulink è un ambiente grafico che consente di descrivere e simulare sistemi complessi. La simulazione di un sistema dinamico con Simulink consiste in due fasi fondamentali:

1. La prima fase consiste nella creazione di un modello grafico del sistema da simulare. Questo modello descriverà le relazioni matematiche esistenti tra gli ingressi e le uscite del sistema.
2. La seconda fase consiste nell'utilizzare l'ambiente Simulink per simulare il comportamento del sistema durante una sua evoluzione temporale su un arco di tempo stabilito dall'utente.

Di fatto Simulink utilizza le informazioni contenute nel modello grafico, che l'utente deve specificare nella fase di descrizione, per generare le equazioni dinamiche del modello e risolvere numericamente il problema di simulazione in esame. Simulink interagisce direttamente con il Workspace di MATLAB. I modelli descritti, pertanto, potranno contenere variabili contenute nel Workspace della sessione corrente. Allo stesso modo, i risultati della fase di simulazione possono essere passati direttamente al Workspace sotto forma di nuove variabili, pronte per essere analizzate. I risolutori che Simulink offre per poter risolvere i sistemi, si suddividono in due famiglie: i metodi di integrazione a passo variabile (variable step); i metodi di integrazione a passo fisso (fixed step). Entrambi i metodi contengono a loro interno numerose integrazioni.

ode45	metodo di Runge-Kutta(4,5)
ode23	metodo di Runge-Kutta(2,3)
ode113	metodo di Adams-Bashforth-Moulton PECE.
ode15s	metodo che utilizza formule di differenziazione, da utilizzare in presenza di sistemi rigidi (stiff problems)
ode23s	metodo modificato di Rosenbrock per sistemi rigidi
discrete	metodo a passo variabile per sistemi puramente discreti

ode5	versione a passo fisso del metodo ode45
ode4	metodo di Runge-Kutta di 4. ordine
ode3	versione a passo fisso di ode23
ode2	metodo di Heun
ode1	metodo di Eulero
discrete	versione a passo fisso per sistemi puramente discreti

Figura 20-*a) variable step; b) fixed step*

2.2 System Generator

System Generator è un'ambiente di programmazione e sviluppo per sistemi real-time, che permette di programmare dispositivi hardware, realizzando un sistema flessibile direttamente nelle programmazione ad alto livello. Permette anche ai progetti di essere composti da una varietà di ingredienti, come i modelli di flusso di dati, i linguaggi tradizionali di progettazione hardware (VHDL, Verilog) e le funzioni che derivano dal linguaggio di programmazione MATLAB, che possono essere utilizzati fianco a fianco, e sintetizzati in hardware di lavoro. I risultati della simulazione System Generator hanno la caratteristica di essere a *bit-accurate* e a *cycle-accurate* ciò significa che i risultati osservati in simulazione corrispondono esattamente ai risultati che si vedono in hardware. Le simulazioni risultano notevolmente più veloci di quelle dei simulatori HDL tradizionali, ed i risultati sono più facili da analizzare.

2.2.1 Blocksets

I *blocksets* che vanno a costituire il progetto, non solo sostituiscono i componenti hardware, ma rispondono al ambiente di progettazione, regolando automaticamente i risultati che vengono prodotti. Tanto che per la modellazione del progetto, i *System Generator blocksets* vengono utilizzati come i *Simulink blockset*, e si trovano nel *Simulink block editor*. (un esempio di libreria è riportato nella *Figura 21*)

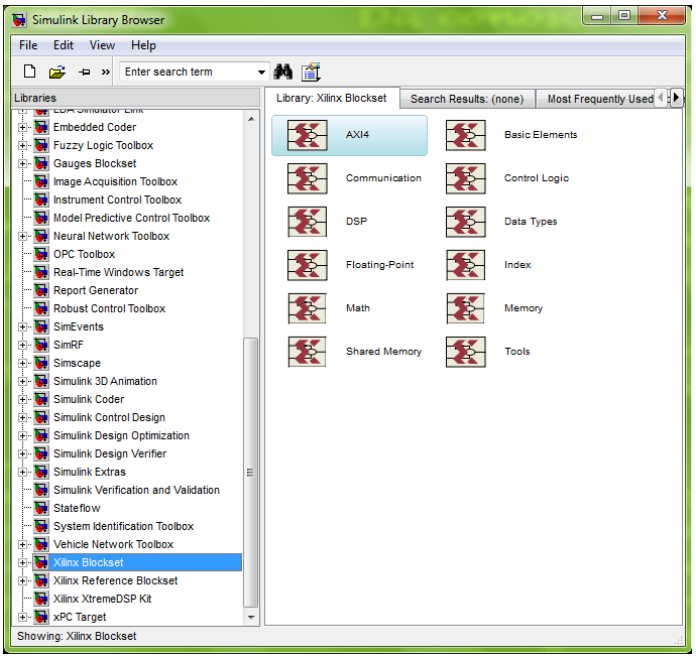


Figura 21-*System Generator blocksets*

I blocchi forniscono astrazioni di funzioni matematiche, logiche, memoria e DSP che possono essere utilizzate per costruire sofisticate elaborazioni del segnale e/o di altri sistemi. Sono presenti anche blocchi che fungono da interfacce ad altri strumenti software (per esempio, FDATool, ModelSim) ed anche il blocco *System Generator* per la generazione del codice che verrà descritto in seguito. Le due più grandi famiglie di *blocksets*, che si utilizzano, sono: *Xilinx Blockset* che contiene i blocchi di base, avente alcuni blocchi che sono di basso livello, ed altri che sono di alto livello (*Figura 22*); *Xilinx Reference Blockset* che contiene i *composite System Generator blocks*, infatti ogni blocco in questo *blockset* è una struttura mista, cioè che la si può implementare come un sottosistema mascherato, con parametri di configurazione del blocco.

Figura 22-*Xilinx Blockset*

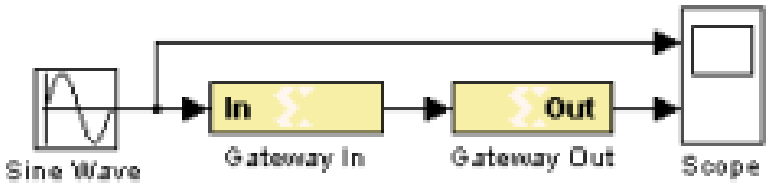
2.2.2 Tipi di segnali

Nell' ambiente Xilinx System Generator i segnali non sono solo bit, ma possono essere numeri in virgola fissa con segno o senza segno, e le modifiche al progetto si traducono automaticamente in opportune modifiche nel tipo di segnale. Al fine di fornire simulazioni *a bit-accurate* in hardware, *blocksets* elaborano il segnale a virgola fissa di precisione booleano e arbitraria. Essendo però implementati i blocchi in ambiente Simulink che lavora di default con segnali a *double floating point*, il collegamento tra i

blocchi Xilinx e blocchi non Xilinx è garantito dai blocchi di *Gateway*. Il blocco *Gateway* converte un segnale in *double floating point* in un segnale Xilinx, e viceversa , con i blocchi *Gateway In* e *Gateway Out*. Anche se la maggior parte dei blocchi Xilinx sono polimorfici, cioè, sono in grado di dedurre opportuni tipi di uscita in base alle loro tipi di ingresso, si può impostare il tipo di uscita per un blocco specificando come deve essere gestita la quantizzazione. La possibilità per un blocco di quantizzare include arrotondamento imparziale verso più o meno infinito, a seconda del segno, o troncamento. Le opzioni di overflow includono saturazione e troncamento.

Inoltre le simulazioni di System Generator sono *bit-true* e *cycle-true modeling.* Una simulazione è *bit-true*, quando ai "confini" il valore prodotto nella simulazione è bit per bit identico al valore corrispondente prodotto in hardware. Una simulazione è *cycle-true* quando ai "confini" i valori prodotti corrispondono a tempi reali. I "confini" della progettazione sono i punti in cui esistono i blocchi System Generator di Gateway, infatti quando un progetto è tradotta in hardware, *Gateway In* (rispettivamente, *Gateway Out*) diventa una porta di alto livello di ingresso (rispettivamente, di uscita).

Progetti di System Generator sono sistemi a tempo discreto. In altre parole, i segnali e i blocchi che li producono hanno associato frequenze di campionamento. Frequenza di campionamento di un blocco determina la frequenza con cui il blocco viene svegliato (permettendo al suo stato di aggiornare). Un modello semplice che mostra il comportamento dei sistemi a tempo discreto, è quello in *Figura 23*; esso contiene un gateway che viene pilotato da un generatore Simulink (onda sinusoidale). Il blocco Gateway In è configurato con un periodo di campionamento di un secondo. Il blocco di Gateway Out converte il segnale da *fixed point* a *double floating point* (in modo che possa essere analizzato nell'ambiente Simulink), ma non viene cambiata la frequenza di campionamento. L'uscita portata nel Scope di Simulink, mostra la versione inalterata e quella campionata dell'onda sinusoidale .



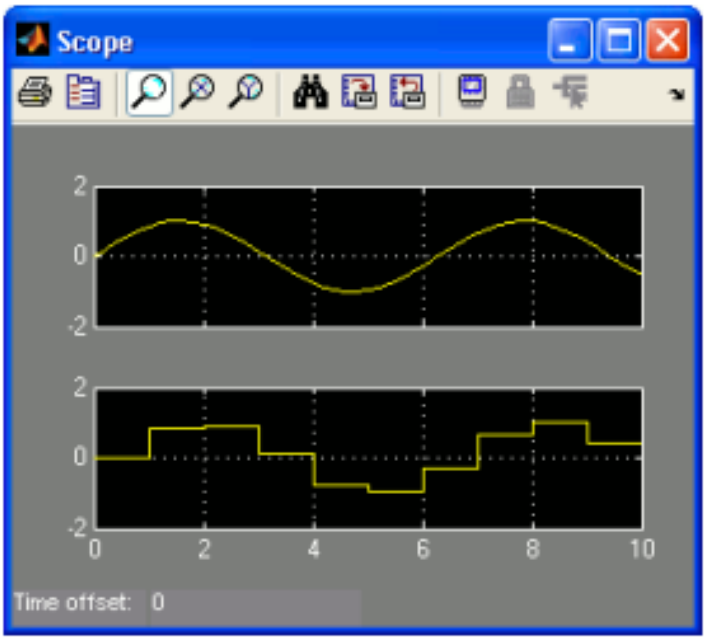


Figura 23-Modello di utilizzo dei blocchi Xlinx System Generator

2.2.3 Multifase

System Generator supporta anche progetti multifase, vale a dire, i progetti che hanno segnali che funzionano in diversi frequenze di campionamento e li compila automaticamente nell'hardware. In questo modo i progetti risultano attuabili in un modo naturale e semplice per Simulink.

Va sottolineato che alcuni *blocksets* possono essere sovracampionati, vale a dire che la loro elaborazione interna viene effettuata a una frequenza maggiore rispetto alla loro velocità di trasmissione dati. In hardware, ciò significa che il blocco richiede più di un ciclo di clock per elaborare un campione di dati. In Simulink tali blocchi non hanno un effetto osservabile su frequenze di campionamento, infatti i blocchi che vengono sovracampionati non causano una variazione di frequenza di campionamento esplicita, ma System Generator considera la frequenza del blocco interno, insieme a tutte le altre frequenze di campionamento quando si va alla generazione del clock logico per l'implementazione hardware. Ciò significa che è necessario prendere in considerazione la frequenza di elaborazione interna dei blocchi sovracampionati quando si specifica il valore del periodo del sistema Simulink nella finestra di blocco di dialogo Generatore di sistema.

2.2.4 Metodi di compilazione

La compilazione a basso livello del progetto viene fatta automaticamente in System Generator. I modi in cui compila un modello possono variare e dipendono dalle

impostazioni nel blocco System Generator. Oltre a produrre descrizioni HDL di hardware, lo strumento genera file ausiliari. Alcuni file (ad esempio, i file di progetto, file vincoli) assistono gli strumenti a valle, mentre altri (ad esempio, VHDL testbench) sono utilizzati per la verifica della progettazione.

I progetti sono compilati e simulati utilizzando il blocco di System Generator, si veda figura. Prima che un progetto (System Generator) possa essere simulato o tradotto in hardware, esso deve includere un blocco System Generator. Quando si crea un nuovo modello, è una buona idea aggiungere immediatamente un blocco System Generator. Premendo il pulsante *Generate,* System Generator si incarica di compilare in modello o la parte del modello in un linguaggio a basso livello equivalente.

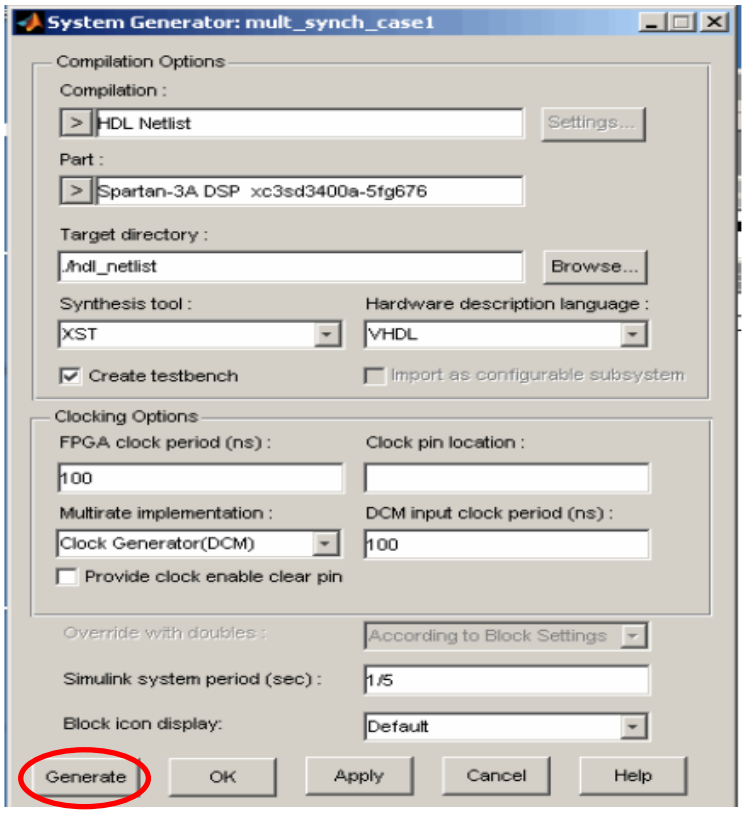


Figura 24-*Schermata pulsante Generate*

Il tipo di compilazione specifica il tipo di risultato che deve essere prodotto. I possibili tipi sono:

- Due tipi di *netlists*: *HDL Netlist* e *NGC Netlist*. *HDL Netlist* è il tipo utilizzato più spesso. In questo caso, il risultato è una raccolta di file HDL, e alcuni file ausiliari che semplificano lavorazione a valle. La raccolta è pronta per essere trasformata con un strumento di sintesi, in una sequenza di istruzioni per la

configurazione, per esempio, di un FPGA. I file che vengono realizzati sono descritti più dettagliatamente per la compilazione. *NGC Netlist* è simile a *HDL Netlist* ma i file risultanti sono file NGC anziché i file di HDL;

- *Bitstream* produce una sequenza di configurazione FPGA che è pronta per essere eseguita in una piattaforma hardware FPGA. Con questa compilazione FPGA è in grado di partecipare alle simulazioni in Simulink;
- *strumento di esportazione EDK*, per esportare la Xilinx Embedded Development Kit in diverse varietà di hardware co-simulazione;
- *Analisi Timing* - una relazione sui tempi del progetto.

Oltre alla possibilità di essere letti da altri linguaggi standard, come già detto in precedenza, è possibile attraverso il blocco *Black Box* introdurre nella simulazione altri questi linguaggi standard (VHDL, Verilog) che possono andar a comporre parte della simulazione.

2.3 Esempi di Utilizzo

Di seguito vengono riportati, nell'ambito dell'elettronica di potenza, degli esempi fatti in Xilinx System Generator. Sono esempi in cui si va realizzare la modellistica per la simulazione real-time che è il cuore per poter poi lavorare con Hardware-in-the-Loop.
Negli esempi, i *System Generator blockset* avranno come segnali di ingresso nel caso dei filtri di primo ordine un generatore Simulink in cui la frequenza incrementa in modo lineare con il tempo, mentre per gli inverter si userà un generatore Simulink con frequenza fissata. Nell'ultimo esempio si userà un generatore Simulink d'onda quadra. Si farà girare tutte simulazioni della parte discreta con un tempo di campionamento intorno a 1μs o anche più piccolo visto che normalmente è il tempo con i quale girano gli FPGA per avere un'apprezzabile quantizzazione del segnale.

2.3.1 Filtro Passa Basso

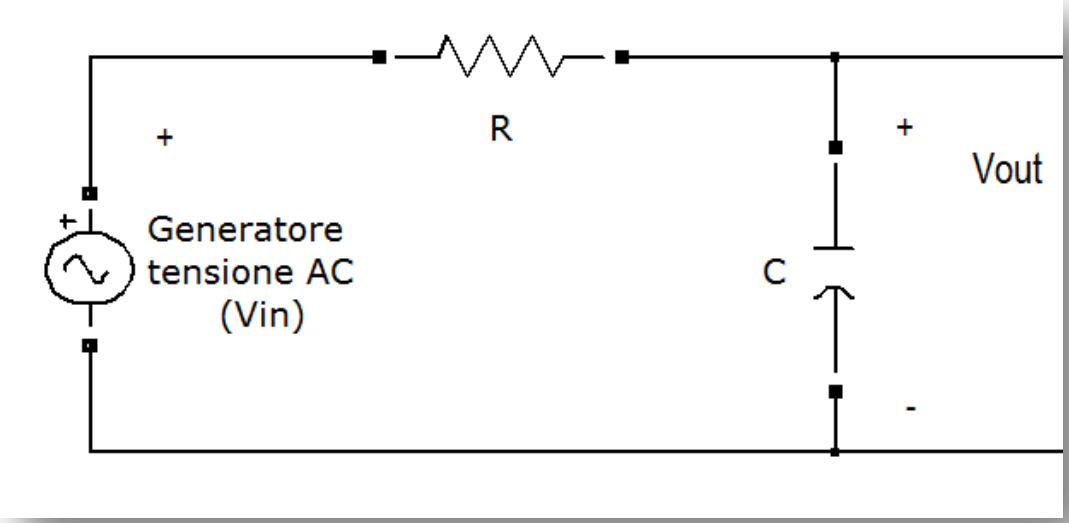


Figura 25-*Filtro passa basso RC*

Si prende in considerazione, un filtro del primo ordine, un filtro passa basso RC.
Si analizza il sistema costituito da un generatore di tensione alternata $(v_{in}(t))$, e da due bipoli passivi: una resistenza (R) e da un condensatore (C). Si ricava un'equazione rispetto alla tensione di uscita ai capi del condensatore:

$$v_{out}(t) = -R * i_R(t) + v_{in}(t) = -R * i_C(t) + v_{in}(t) = -RC * \frac{dv_C(t)}{dt} + v_{in}(t)$$

e si ottiene come funzione di trasferimento $W(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{(1+\tau s)}$ *(1)*

con $\tau = \mathrm{RC}$.
La quale servirà per andar a costruire il schema discreto che sarà implementato dai blocchi Xilinx System Generator.
Si osserva il modulo della funzione di trasferimento; c'è un punto di spezzamento a $\tilde{\omega} = \left|\frac{1}{\tau}\right|$ ed ha una banda passante (a -3dB) di $B_3 = \frac{1}{2\pi*\tau}$. *Nella Figura 26* viene riportato un diagramma di Bode con valori delle variabile che si prenderà successivamente in analisi.

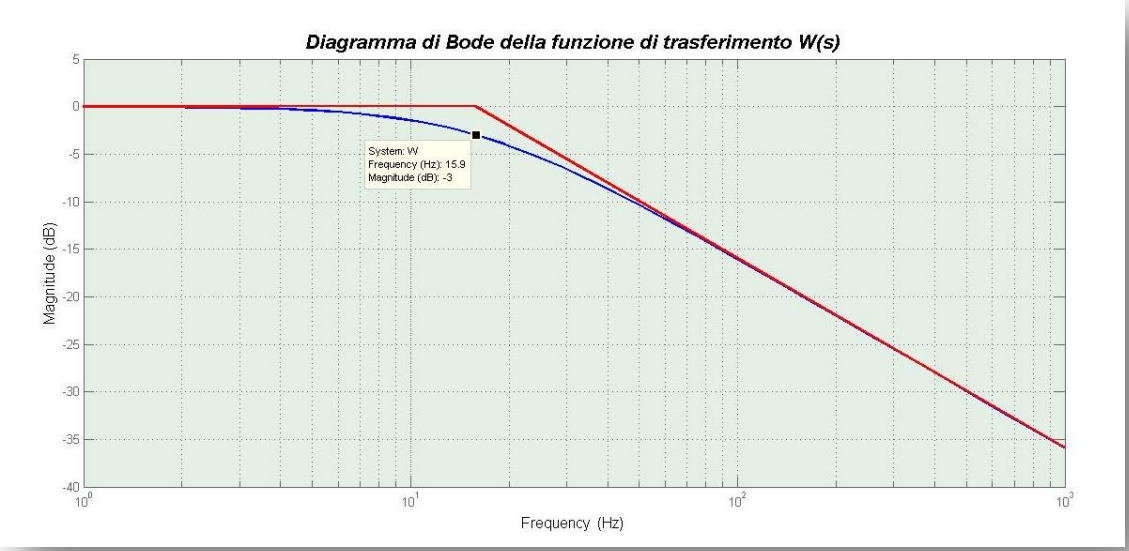


Figura 26-*Diagramma di Bode reale(rosso) e asintotico(blu)*

Per poter simulare il filtro si fa uso del software Matlab-Simulink. Per aver un utilizzo più comodo, ma soprattutto più efficiente delle variabili che serviranno nella simulazione, si crea un file di tipo m, che permette di essere letto da Matlab e di interagire con la simulazione attraverso la "Workspace" di Matlab. Nel file si definisce il valore della resistenza $(R = 1e^4\ Ohm)$, del condensatore $(C = 1e^{-6}\ F)$ e della costante τ (riportato il codice nell'Appendice A). In Simulink si va a imporre come tipo di risolutore un solver a passo fisso (di tipo Eulero).Per progettare il filtro attraverso Xilinx System Generator, come prima cosa si riprende le formule calcolate precedentemente che devono essere discretizzate, ciò è possibile con l' utilizzo della trasformata bilineare o di Tustin, che permette di lavorare nelle trasformate Zeta a tempo discreto. La trasformata di Tustin dice che $s = \frac{2}{T}\frac{z-1}{z+1}$. Andando a sostituire s dalla funzione di trasferimento *(1)* si ottiene che

$$W(z) = \frac{1}{1+\tau\left(\frac{2}{T}\frac{z-1}{z+1}\right)} = \frac{z+1}{\left(1+\frac{2\tau}{T}\right)z+\left(1-\frac{2\tau}{T}\right)}$$

quindi:

$$W(z) = \frac{T}{2\tau+T}\frac{1+z^{-1}}{1+\left(\frac{T-2\tau}{T+2\tau}\right)z^{-1}} = b_0\frac{1+b_1z^{-1}}{1+a_0z^{-1}}$$

Con $b_0 = \frac{T}{2\tau+T}$; $b_1 = 1$; $a_0 = \frac{T-2\tau}{T+2\tau}$;.

Si calcola b_0, b_1 e a_0 per poterli andar inserire i valori nei blocchi Xilinx System Generator. Per poter scegliere un valore del periodo di campionamento adeguato bisogna sottostare a due condizioni, che $f_c > 2 * f_m$ cioè che la frequenza di campionamento sia almeno due volte maggiore alla frequenza massima dello spetro del segnale da campionare; che $f_N \gg B_3$ che la frequenza di Nyquist sia molto maggiore alla banda passante e, questo comporta ad avere il tempo di campionamento che sia qualche ordine di grandezza più piccolo della banda passante. I tools che vengono utilizzati per costruire il modello sono quelli sommatori o sottrattori (AddSub), quelli per moltiplicare delle costanti (CMux) e quelli che fanno da ritardatore nella trasformata z (Delay). Ma affinché il sistema discreto funzioni servono i tre blocchetti fondamentali In/Out Gateway e System Generator.

Il sistema ottenuto è formato da un segnale di ingresso il quale viene moltiplicato per una costante (b0) il segnale entra in un sistema a retroazione non unitario, viene moltiplicato per il ritardatore e nel tratto di anelo chiuso viene moltiplicato per una costante (a0) dopo di che il segnale viene moltiplicato per un altro ritardatore ad anello aperto moltiplicata per un'altra costante (b1). Schema riportato nella *Figura 27* di seguito.

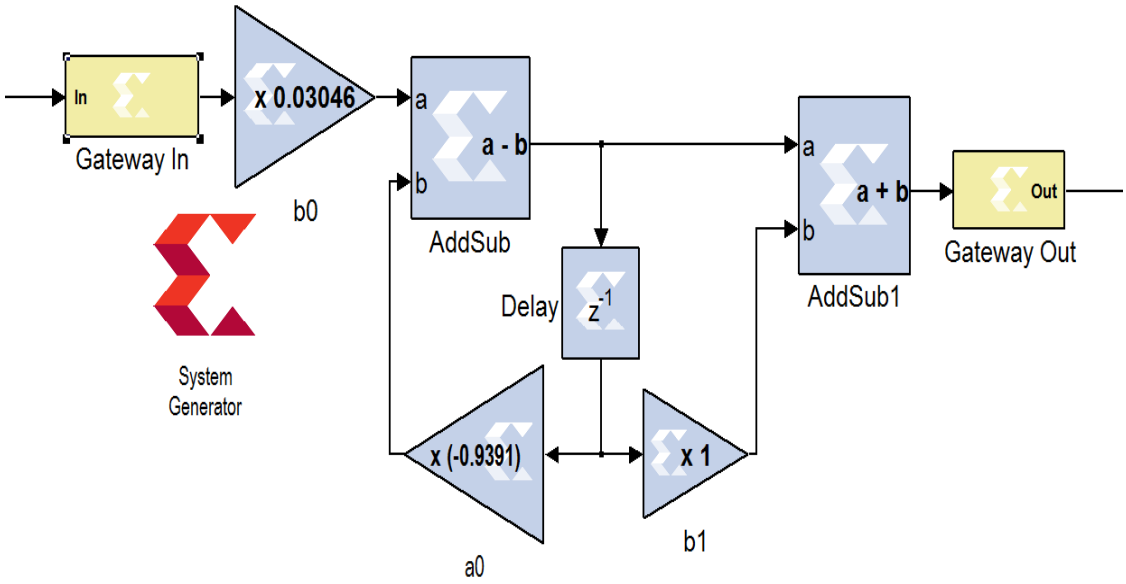


***Figura 27**-Schema a blocchi del filtro a tempo discreto.*

Facendo andare la simulazione si vede che:

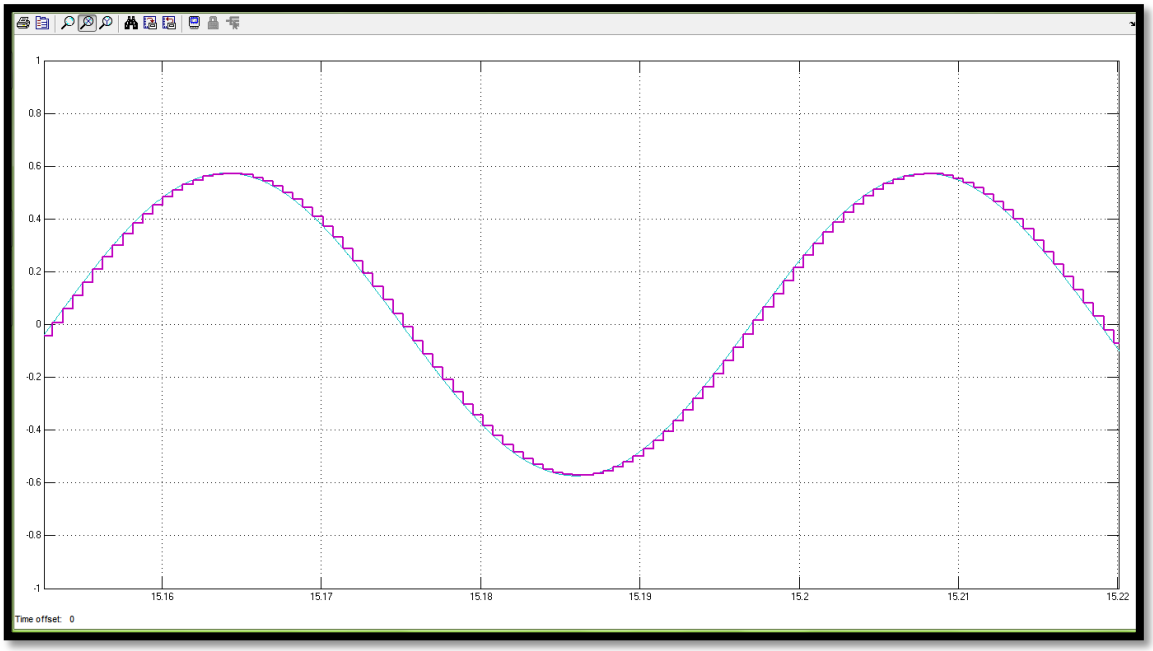


***Figura 28**-Sinusoide azzurra è il segnale filtrato a tempo continuo quella viola è quello discretizzato*

La figura riportata raffigura il segnale discretizzato da Xilinx System Generator sovrapposto al segnale filtrato ottenuto a tempo continuo grazie ai blocchi presenti nella libreria di Simulink. Si riesce ad ottenere un segnale a tempo discetto molto accettabile. Più si va a ridurre il tempo di campionamento maggiore è la precisione del segnale, ciò comporta però ad avere un strumento reale che riesca poi a lavorare a quel tempo.

Si vede che per valori di frequenza del segnale di ingresso molto minori di B_3, il sistema non induce nessuna significativa attenuazione e nessuna significativa variazione di fase, mentre, al crescere della frequenza di ingresso, l'attenuazione e la variazione di fase diventano non trascurabile. In particolare al divergere della frequenza il filtro tende a "bloccare " completamente il segnale di ingresso .

Se si utilizzasse un valore di frequenza di campionamento pari a $\mathrm{f_c} = 100\ \mathrm{Hz}$ fin che si ha valori di frequenza del segnale di ingresso bassi non si ha problemi di campionamento del segnale, mentre non appena inizia ad aumentare la frequenza inizia a sorgere dei problemi di campionamento questo è causato perché non vengono più rispettate tutte le condizioni in particolare quella di avere un una frequenza di Nyquist molto più grande della banda passante del sistema a catena chiusa. Bisogna però ricordare che i dispositivi reali con periodo di campionamento molto piccolo hanno un elevato costo.

2.3.2 Filtro Passa Alto

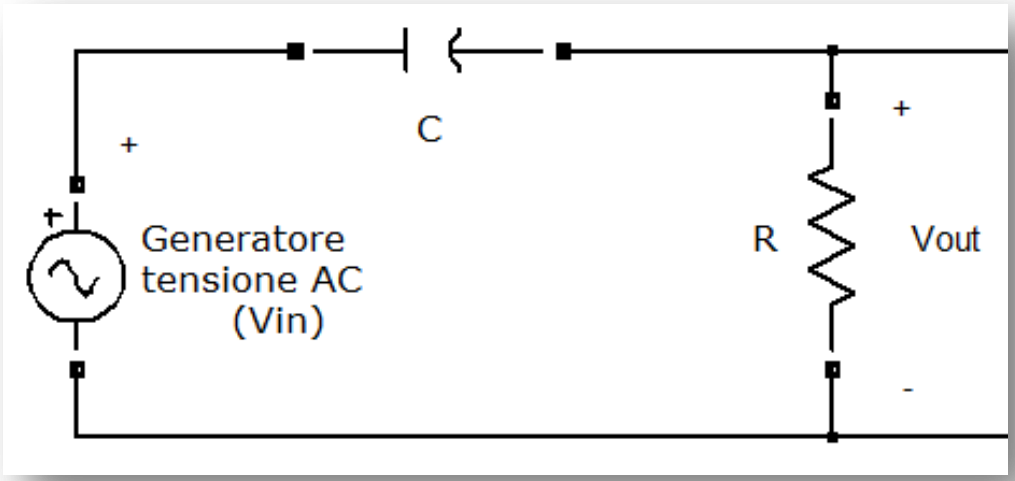


Figura 29-*Filtro passa alto RC*

Si prende in considerazione, come filtro del primo ordine, un filtro passa alto RC
Si analizza il sistema costituito da un generatore di tensione alternata $(v_{in}(t))$, e da due bipoli passivi: un condensatore (C) e da una resistenza (R).
Si ricava un'equazione rispetto alla tensione di uscita ai capi della resistenza:

$$v_{out}(t) = RC * \frac{dv_C(t)}{dt}$$

si pone $\tau = \mathrm{RC}$. Si ha che: $V_{out} = \frac{\tau s * V_{in}}{(1+\tau s)}$

e si ottiene come funzione di trasferimento $W(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{\tau s}{(1+\tau s)}$ *(2)*

Si osserva il modulo della funzione di trasferimento, che c'è un punto di spezzamento a $\tilde{\omega} = \left|\frac{1}{\tau}\right|$ ed ha una banda passante (a -3dB) di $B_3 = \frac{1}{2\pi * \tau}$; nella *Figura 30* viene riportato un diagramma di Bode con valori delle variabile che si prenderà successivamente in analisi.

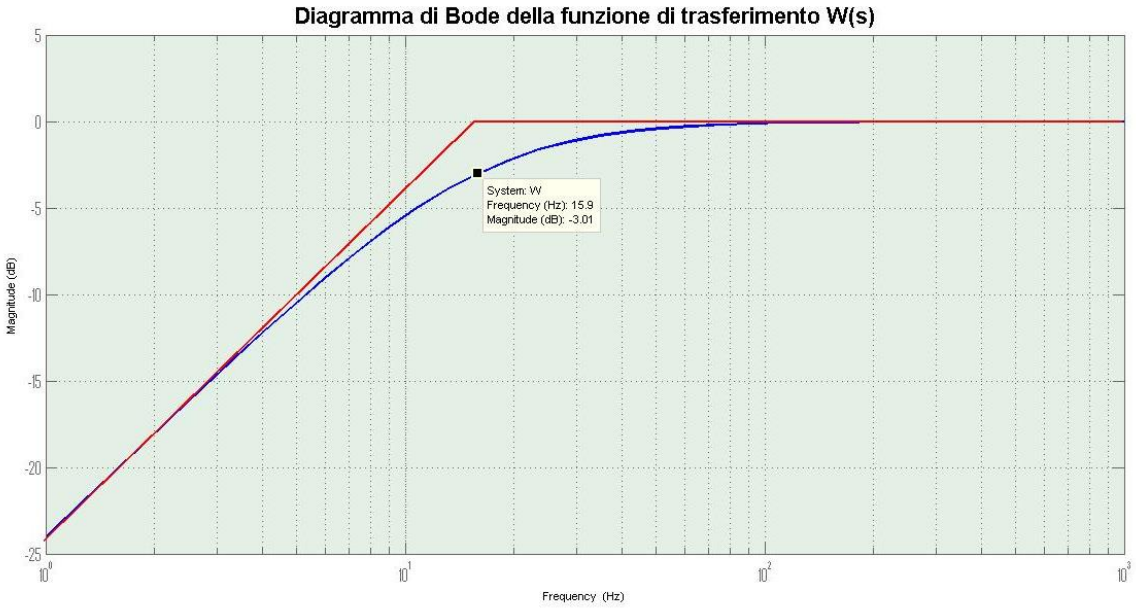


***Figura 30**-Diagramma di Bode reale(rosso) e asintotico(blu)*

Per poter simulare a tempo reale il filtro precedentemente analizzato si fa uso del software Simulink. Per aver un utilizzo più comodo, ma soprattutto più efficiente delle variabili che serviranno nella simulazione, si va a creare un file di tipo m, che permette di essere letto da Matlab e di interagire con la simulazione attraverso la "Workspace" di Matlab. Nel file si va a definire il valore della resistenza $(R = 1e^4\ Ohm)$, del condensatore $(C = 1e^{-6}\ F)$, la costante τ (*si veda appendice A*). In Simulink si va a imporre come tipo di risolutore un solver a passo fisso (di tipo Eulero). Per progettare il filtro attraverso Xilinx System Generator come prima cosa riprendendo le formule calcolate precedentemente e come nell'esempio precedente le discretizziamo attraverso la trasformata bilineare o di Tustin, ottenendo dalla funzione di trasferimento *(2)*:

$$W(z) = \frac{2\tau}{T+2\tau}\frac{1-z^{-1}}{1+\left(\frac{T-2\tau}{T+2\tau}\right)z^{-1}} = b_0\frac{1+b_1z^{-1}}{1+a_0z^{-1}}$$

Con $b_0 = \frac{2\tau}{T+2\tau}$; $b_1 = -1$; $a_0 = \frac{T-2\tau}{T+2\tau}$;.

Analogamente all'esempio valgono le considerazioni fatte precedente sia per la scelta del tempo di campionamento sia nella realizzazione dello modello in Xilinx System Generator, che lo si può vedere nella figura seguente

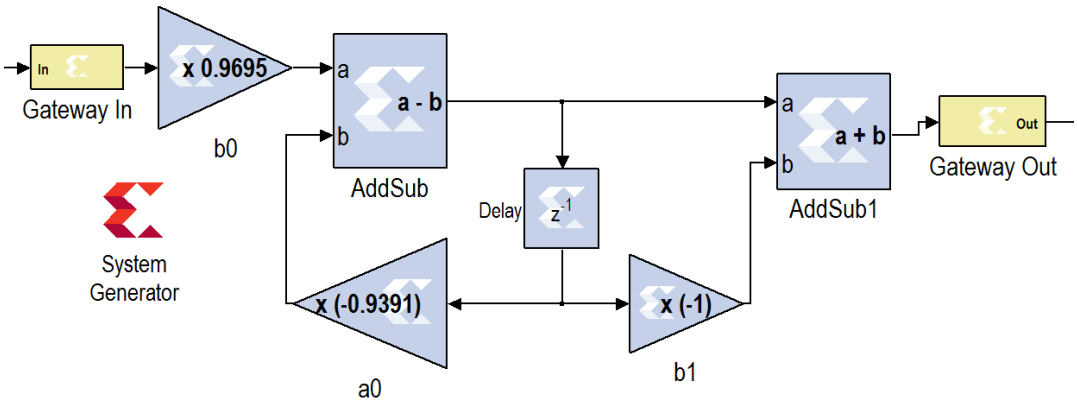


Figura 31-*Schema a blocchi del filtro discretizzato*

Facendo andare la simulazione si vede che:

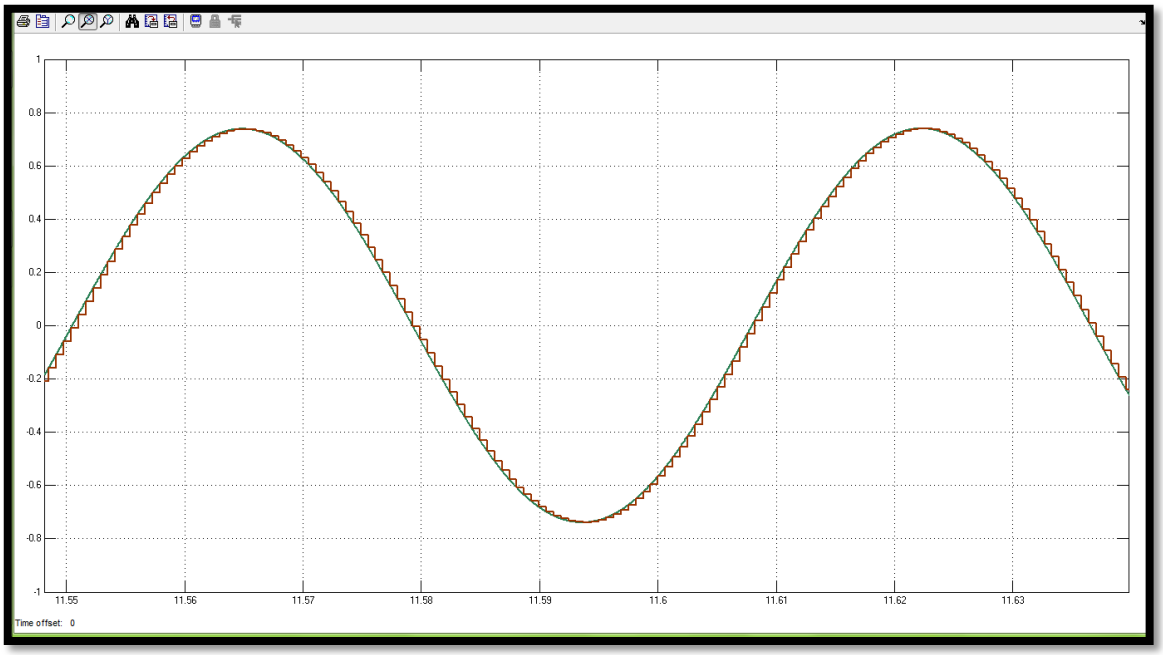


Figura 32-*Sinusoide verde è il segnale filtrato a tempo continuo quella rossa è quello discretizzato*

Il filtro riesce ad attenuare abbastanza bene le frequenze inferiore alla sua frequenza di taglio $f_t = 15{,}91\ Hz$. A riguardo del periodo di campionamento emergono gli stessi problemi riscontrati nell'analisi del filtro passa basso.

2.3.3 Inverter a mezzo ponte

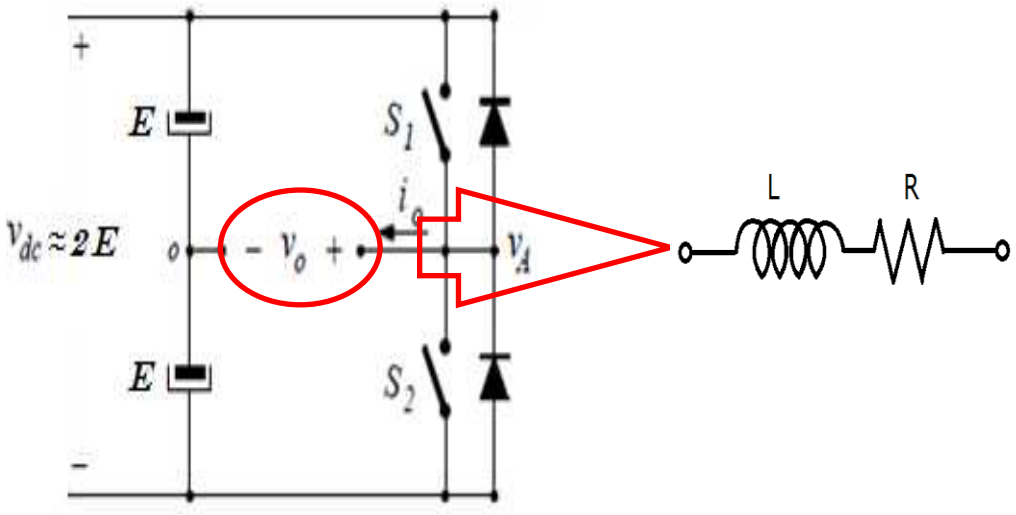


Figura 33-*Inverter a mezzo ponte (half-bridge)con carico ohmico-indittivo*

Questa configurazione è detta inverter a mezzo ponte (half-bridge); essa è formata da due interruttori connessi in serie (composti da dispositivi attivi come, BJT, IGBT), ciascuno dei quali munito di un diodo in antiparallelo in modo da rendere l'interruttore unidirezionale in tensione e bidirezionale in corrente. Il carico è connesso tra i punti intermedi dei due rami. La serie degli interruttori è connessa in parallelo alla serie di due condensatori uguali, la cui capacità deve essere elevata in modo che la tensione centrale rimanga pressoché costante e pari ad $\frac{V_{dc}}{2} = E$.

Questa struttura permette di ottenere solamente due livelli di tensione in uscita, che grazie alla presenza dei diodi in antiparallelo sugli interruttori, sono indipendenti dal verso della corrente. Il primo livello è ottenibile con la configurazione S_1 chiuso ed S_2 aperto, e la tensione di uscita $v_{out}(t)$ applicata al carico è positiva e pari ad $\frac{V_{dc}}{2}$, mentre con il secondo livello, dato da S_1 aperto ed S_2 chiuso, la tensione di uscita si porta ad una valore negativo pari a $-\frac{V_{dc}}{2}$. I ritardi e le imprecisioni dei comandi possono causare la sovrapposizione della conduzione dei due interruttori che porterebbe ad un cortocircuito quasi sempre distruttivo tra l'alimentazione del lato in continua. Per ovviare a questo tipo di problema c'è l'inserimento di un tempo morto ("*dead time*") per garantire che ciascun interruttore di ramo sia effettivamente aperto quando l'altro è in fase di chiusura.

Attualmente le principali tecniche di modulazione che permettono di regolare la frequenza della tensione di uscita, possono essere quelle a modulazione ad onda quadra o quelle a larghezza di impulso(PWM).

Nell'intento di poter creare una simulazione chiara e con lo scopo di poter andar a costruire uno sistema in cui si possa vedere l'effettiva funzionalità della interconnessione tra Simulink e Xilinx System Generator si è scelto di far alcune semplificazione nella costruzione dell'inverter a mezzo ponte. Si consideri i due interruttori connessi in serie come due semplici interruttori che aprono e chiudono, senza tener conto del "*dead time*" enunciato precedentemente. Si prende in osservazione un inverter a mezzo ponte che abbia come carico ohmico-induttivo, con l'intento di andar a vedere la corrente all'uscita del carico, e allora si consideri:

- $\delta(t)$ quando S_1 chiuso e S_2 aperto

  Si ha che $v_L(t) = L\frac{di_{out}}{dt} = \frac{V_{DC}}{2} - i_{out} * R$;

- $[1 - \delta(t)]$ quando S_1 aperto e S_2 chiuso

  di può vedere che $v_L(t) = L\frac{di_{out}}{dt} = -\frac{V_{DC}}{2} - i_{out} * R$;

se si fa uguaglianza delle due equazioni

$$v_L(t) = L\frac{di_{out}}{dt} = \left(\frac{V_{DC}}{2} - i_{out} * R\right) * \delta(t) + \left(-\frac{V_{DC}}{2} - i_{out} * R\right) * [1 - \delta(t)] =$$

$$= \frac{V_{DC}}{2}\delta(t) - i_{out} * R\delta(t) - \frac{V_{DC}}{2} - i_{out} * R + \frac{V_{DC}}{2}\delta(t) + i_{out} * R\delta(t) =$$

$$= V_{DC}\delta(t) - \frac{V_{DC}}{2} - i_{out} * R.$$

Quindi $L\frac{di_{out}}{dt} + i_{out} * R = V_{DC}\delta(t) - \frac{V_{DC}}{2}$.

Può essere schematizzata come in *Figura 34.*

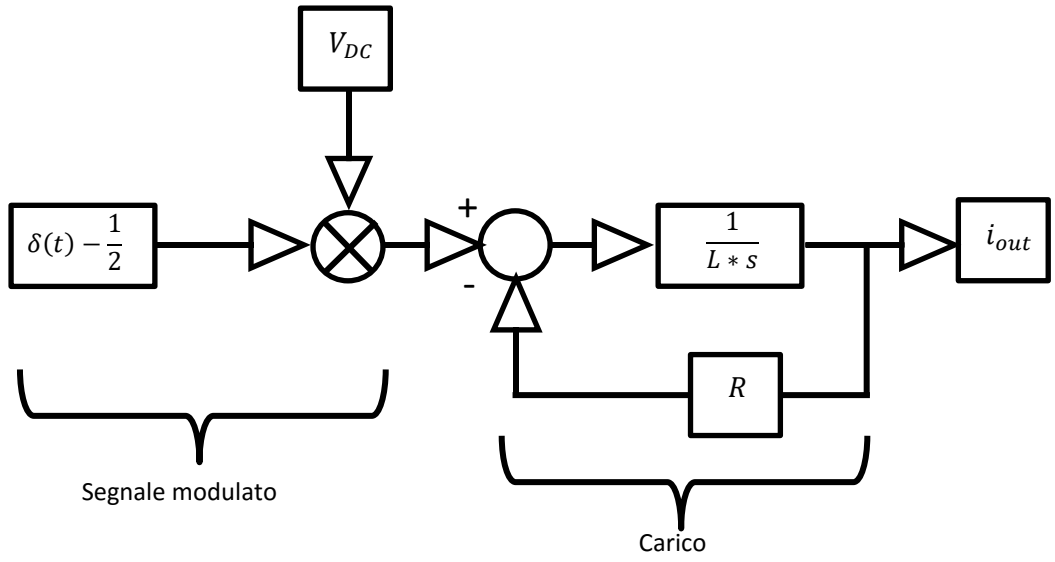


Figura 34-*Schema a blocchi dell'inverter a mezzo ponte*

Per poter creare il segnale modulato pronto per essere successivamente filtrato dal carico, si cerca di riprodurre il metodo della sottoscillazione sinusoidale in cui le inversioni di tensioni di fase di un invertitore sono fatte coincidere con le intersezioni di due terne di segnali di frequenza diversa. Il segnale di frequenza minore è detto *modulante (modulation function)* che nel caso preso in analisi è un segnale sinusoidale con frequenza pari a quella fondamentale $f = 50Hz$ e ampiezza 1; quello di frequenza maggiore è detto *portante (carrier)* che è un segnale triangolare periodico di frequenza di commutazione $f_{ws} = 20e^3\ Hz$ e valori di campo di oscillazione va da -1 a 1. Viene illustrato nella *appendice A* che riporta il file.m di scrittura Matlab. Lo scopo della modulazione a sottooscillazione è di ottenere una tensione che, pur variando tra $\frac{V_{dc}}{2}$ e $-\frac{V_{dc}}{2}$, abbia uno spettro alle basse frequenze identico a quello della modulante. Questa tecnica fa parte della famiglia delle tecniche di modulazione PWM (*pulse width modulation*).

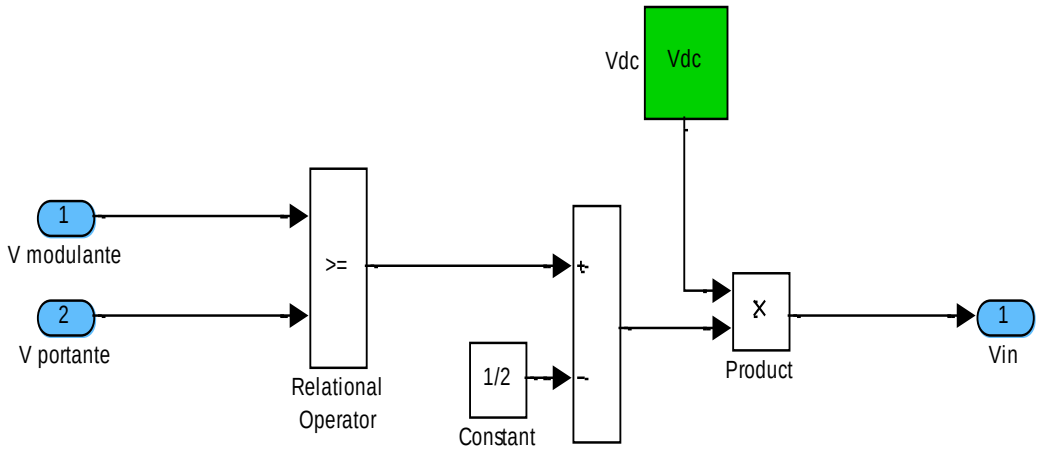


***Figura 35**-Rappresentazione dello schema a blocchi in Simulink del segnale modulante.*

Ottenuto il segnale modulante bisogna spostare l'attenzione al carico. Si osserva la sua funzione di trasferimento:

$P(s) = \frac{1}{Ls+R}$;

Come valori del carico ohmico-induttivo si dà alla resistenza $R = 10\,\Omega$ e all'induttanza $L = 2 * e^{-4}\,H$.

Nella simulazione in Simulink si è impostato una risoluzione a tempo variabile e come tipo di solver "ode45(Dormand-Prince)".Si impone "Max step size" un ventesimo del periodo di commutazione e la durata della simulazione va da 0 a 0.5 secondi.

Si va a discretizzare il carico. Con i pacchetti di System Generator è possibile a farlo, partendo dalla sua funzione di trasferimento la si può discretizzare attraverso la trasformata di Tustin:

se $s = \frac{2}{T}\frac{z-1}{z+1}$ allora

$$W(z) = \frac{1}{\frac{2}{T}\frac{z-1}{z+1} * L + R} = \frac{T(z+1)}{2L(z-1) + TR(z+1)} = \frac{T(z+1)}{2Lz - 2L + TRz + TR}$$

$$= \frac{T(z+1)}{z(2L+TR) + (-2L+TR)} = \frac{T}{TR+2L} * \frac{1+z^{-1}}{1+\left(\frac{TR-2L}{TR+2L}\right)z^{-1}}$$

$$W(z) = b_0 \frac{1 + b_1 z^{-1}}{1 + a_0 z^{-1}}$$

Avendo come valori di:

$b_0 = \frac{T}{TR+2L}$; $b_1 = 1$; $a_0 = \frac{TR-2L}{TR+2L}$.

Si ottiene come schema:

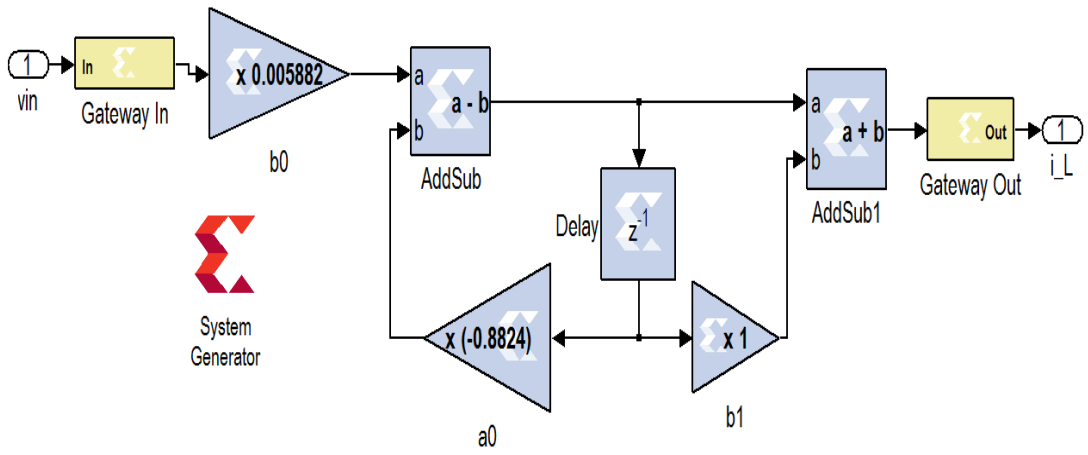


***Figura 36**-Schema del carico in System Generator*

Per poter scegliere un valore del periodo di campionamento adeguato bisogna sottostare a due condizioni, che $f_c > 2 * f_m$ cioè che la frequenza di campionamento sia almeno due volte maggiore alla frequenza massima dello spetro del segnale da campionare; che $f_N \gg B_3$ che la frequenza di Nyquist sia molto maggiore alla banda passante e questo comporta ad avere il tempo di campionamento che sia qualche ordine di grandezza più piccolo della banda passante. Nel caso preso in questione si prende un tempo di campionamento pari a $T = \frac{1}{f_c}$ con $f_c = (fws * 20)\ Hz$ che soddisfa sicuramente tutte le condizioni. Viene riportato il codice Matlab nella *appendice A.* Si ottiene una corrente di uscita che viene illustrata dalla *Figura 37.*

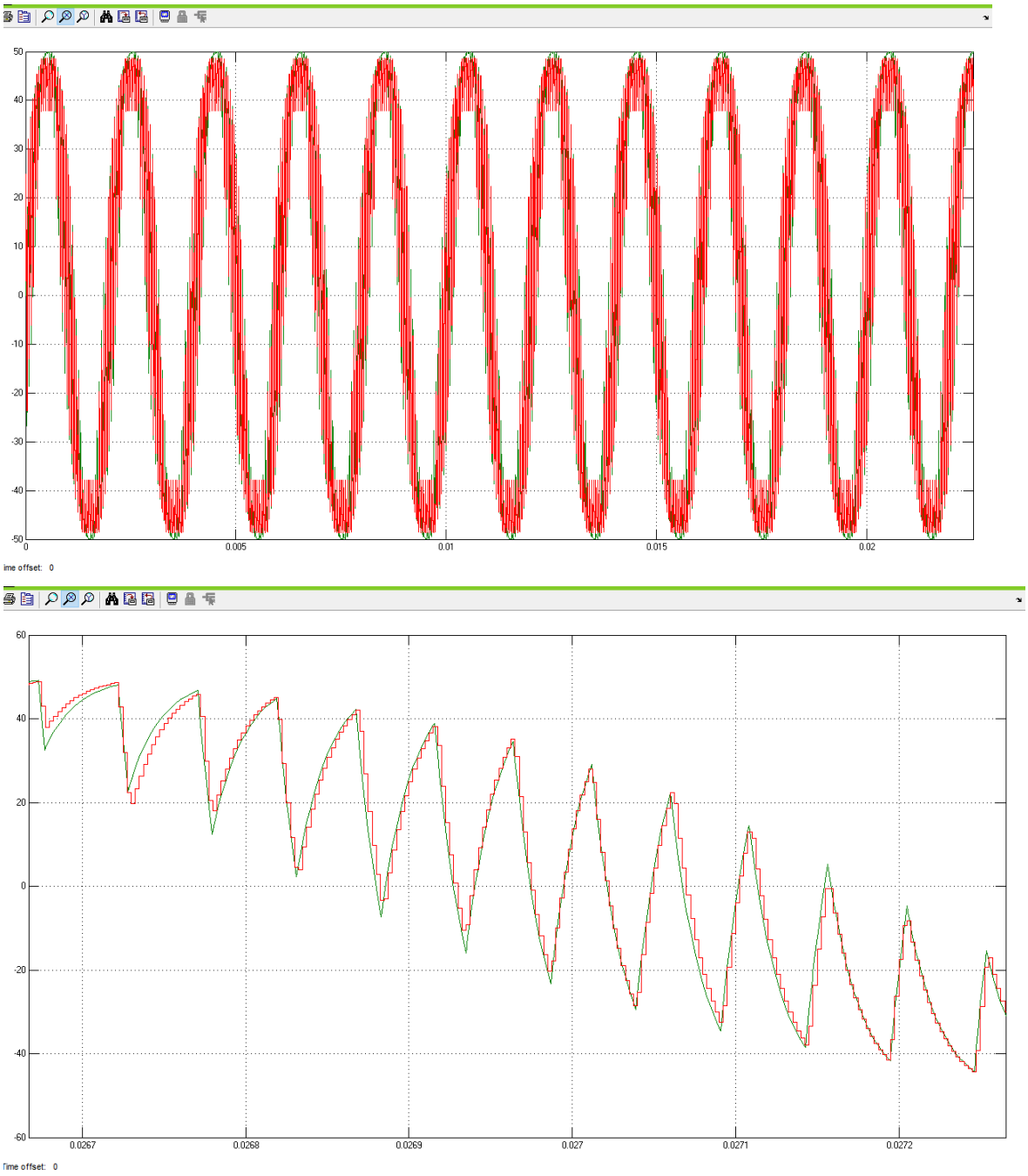


Figura 37-*Diagramma corrente uscita per inverter a mezzo ponte*

2.3.4 Inverter a ponte

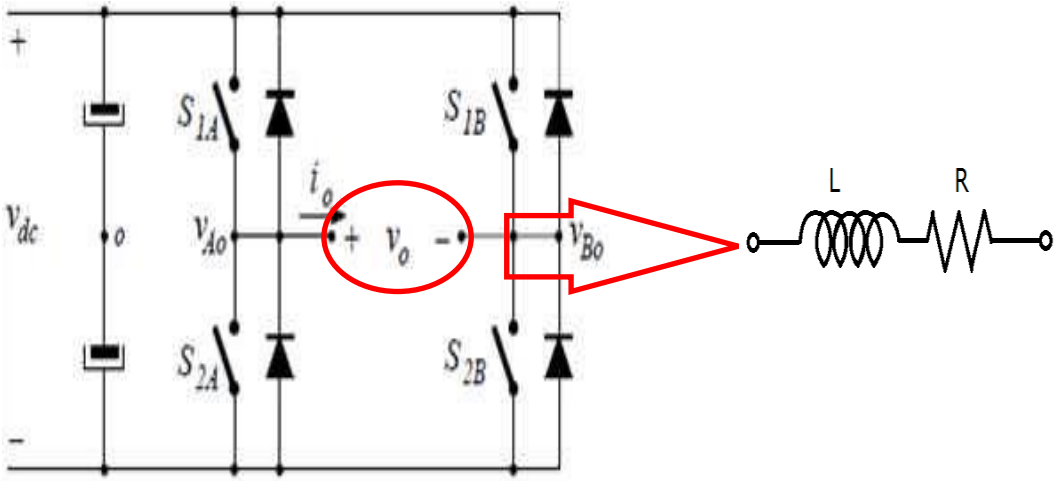


Figura 38-*Inverter a ponte*

Si tratta di uno schema a ponte, formato dalla connessione in serie di due inverter a mezzo-ponte (half-bridge) dove la tensione di uscita totale vale $v_0(t) = v_{A0}(t) - v_{B0}(t)$ e si avrà come segnale modulato un segnale che varia da V_{DC} a $-V_{DC}$. Anche in questo esempio vengono fatte delle semplificazione come quelle per Inverter a mezzo-ponte.

Per poter creare il segnale modulato pronto per essere successivamente filtrato dal carico, si va a prendere a riguardo la tecnica di modulazione PWM (*pulse width modulation*), nella quale gli interruttori diagonalmente opposti (S_{1A} e S_{2B}) e (S_{1B} e S_{2A}) dei due rami della *Figura 38* sono comandati, rispettivamente, come coppie di interruttori 1 e 2. Il segnale di frequenza minore è detto *modulante (modulation function)* che nel caso preso in analisi si usa un segnale sinusoidale con frequenza pari a quella fondamentale di $f = 50Hz$ e ampiezza 1; quello di frequenza maggiore è detto *portante (carrier)* che non è nient'altro che un segnale triangolare periodico di frequenza di commutazione $f_{ws} = 20e^3\ Hz$ e valori di campo di oscillazione da -1 a 1. Si prende in osservazione un inverter a ponte con carico ohmico-induttivo, e allora si consideri:

- $\delta(t)$ quando $S_{1A} - S_{2B}$ chiusi e $S_{2A} - S_{1B}$ aperti

  $v_L(t) = L\frac{di_{out}}{dt} = V_{DC} - i_{out} * R;$

- $[1 - \delta(t)]$ quando $S_{1A} - S_{2B}$ aperti e $S_{2A} - S_{1B}$ chiusi

$$v_L(t) = L\frac{di_{out}}{dt} = -V_{DC} - i_{out} * R;$$

si fa uguaglianza delle due equazioni

$$v_L(t) = L\frac{di_{out}}{dt} = (V_{DC} - i_{out} * R) * \delta(t) + (-V_{DC} - i_{out} * R) * [1 - \delta(t)] =$$

$$= V_{DC}\delta(t) - i_{out} * R\delta(t) - V_{DC} - i_{out} * R + V_{DC}\delta(t) + i_{out} * R\delta(t) =$$

$$= 2 * V_{DC}\delta(t) - V_{DC} - i_{out} * R.$$

Che può essere schematizzata come in *Figura 39.*

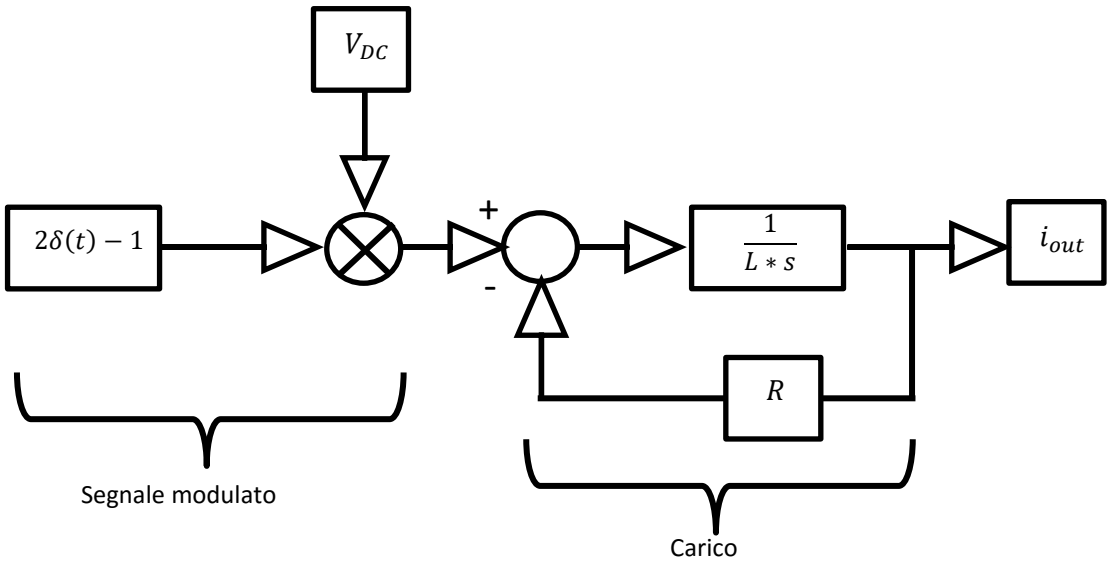


***Figura 39**-Schema a blocchi dell'inverter a ponte*

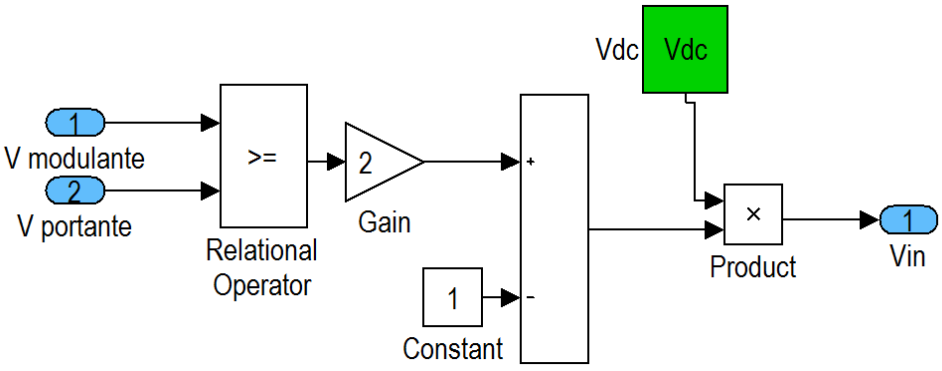


Figura 40-*Rappresentazione dello schema a blocchi in Simulink del segnale modulante.*

La parte del carico viene ripresa interamente quella analizzata nell' inverter a mezzo ponte visto che non va a subire variazioni come schema a blocchi (si vede *Figura 41*).

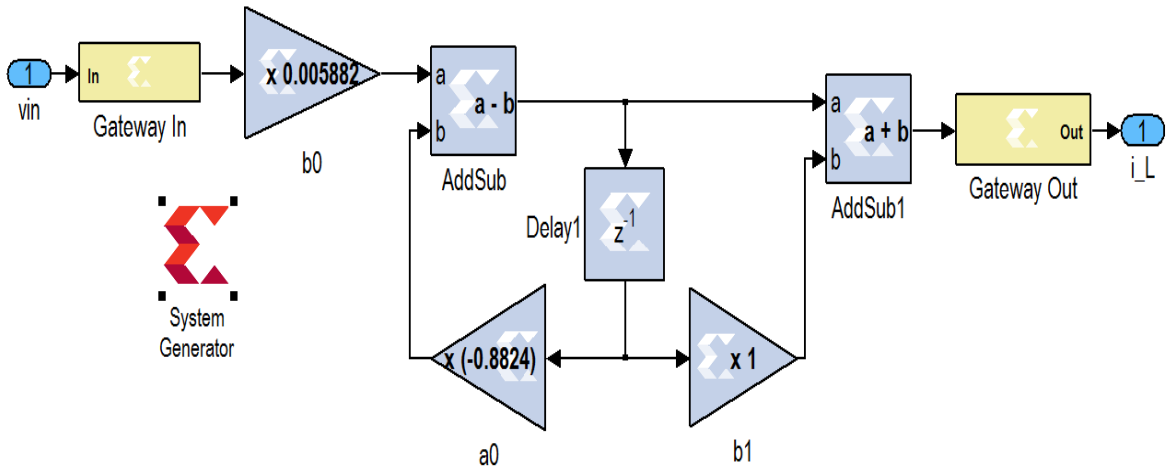
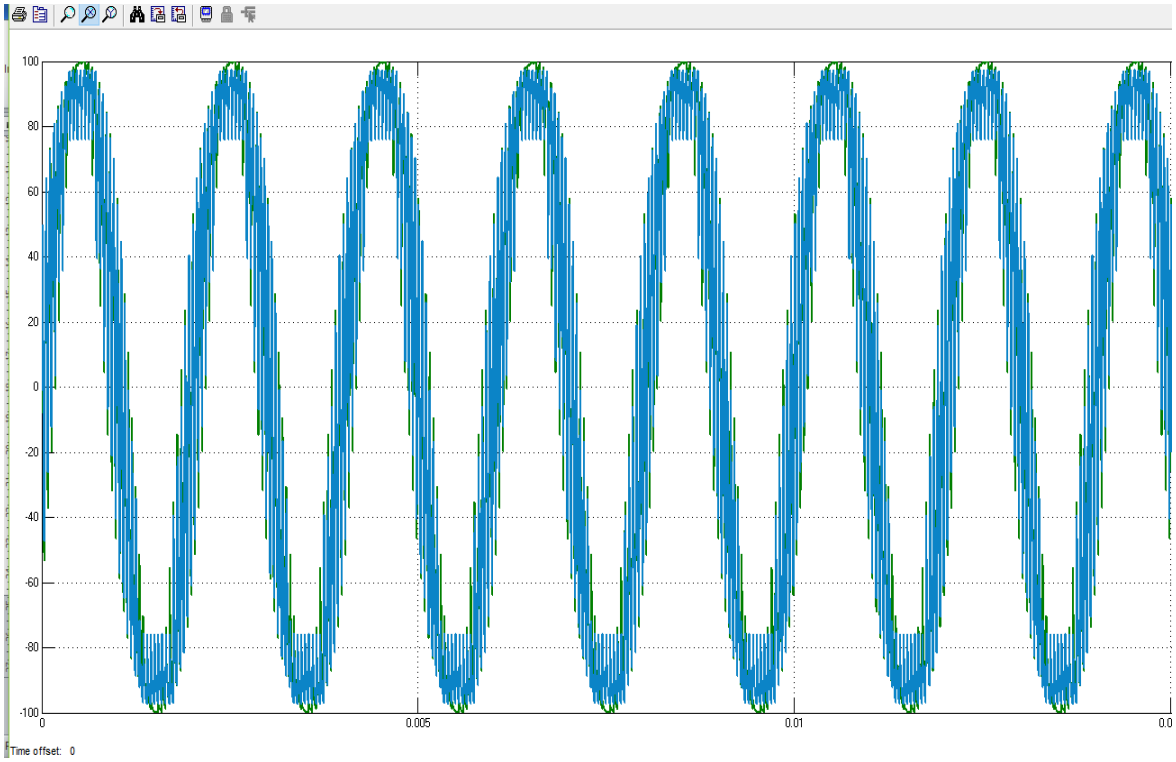


***Figura 41**-Rappresentazione dello schema a blocchi in Simulink*

In *Figura 42* viene riportato l'andamento della corrente per l'inverter a ponte.



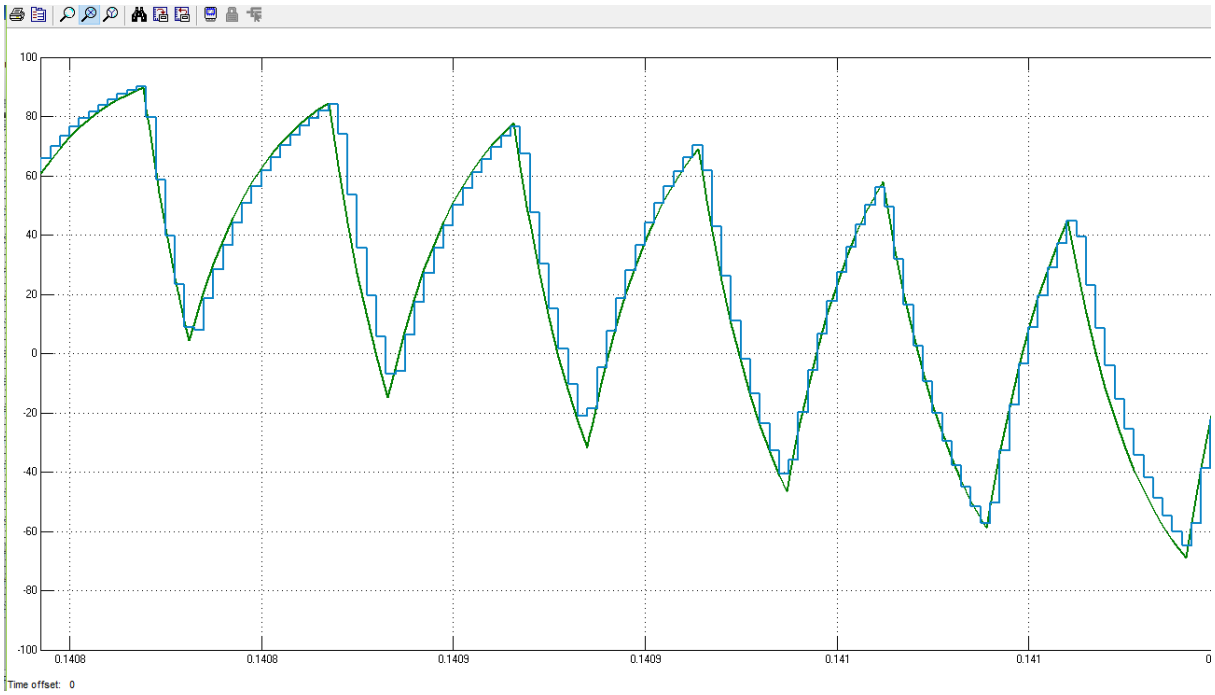


***Figura 42**-Diagramma corrente uscita per inverter a ponte*

Si vede che lavorando con tensione di ingresso costante si riesce ad ottenere in uscita una corrente alternata molto approssimabile a un segnale sinusoidale, se alzo il valore dell'induttanza riesco ad ottenere un segnale più sinusoidale. Con l'inverter a ponte si riesce ad avere in uscita un valore di tensione doppio rispetto a quello dell'inverter a mezzo ponte.

2.3.5 Controllo in retroazione di un Inverter a ponte

In questo ultimo esempio si vorrebbe osservare la realizzazione di un sistema più completo, costituito da un regolatore PI e da un inverter monofase a ponte, avente la parte del carico costruita dai blocchi di Xilinx System Generator, che lavorasse a tempo discreto a differenza di tutto il sistema che lavora a tempo continuo. Il sistema andrebbe a simulare un convertitore controllo di corrente impressa con retroazione unitaria.

Prima di vedere lo schema a blocchi teorico del sistema, si vuole fare delle ipotesi semplificative; si ipotizza di avere segnale iniziale ideale, in questo caso un generatore di corrente avente segnale a gradino nell'istante di tempo di 0.15 sec. e che vada da 0A a 2A. Si ipotizza anche che il regolatore PI non abbia l'errore a regime ad anello ed affinché sia stabile, si abbia un wt che sia 20 almeno volte più piccolo rispetto la frequenza di campionamento e un margine di fase che possa essere tra 50 e 70. Si è scelto un regolatore PI poiché garantisce un' azione regolatrice di buone

caratteristiche, mantenendo a livelli accettabili la complessità della sua realizzazione digitale. Si ipotizzi anche che, sia il sistema che il regolatore, non abbiano ritardi addizionali (legati ai filtro del segnale, al campionamento, ai tempi di elaborazione).
Si ha come possibile schema

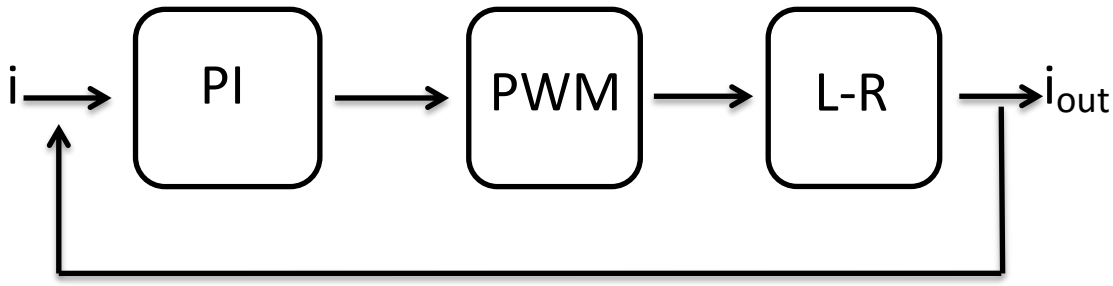


***Figura 43**-Schema teorico del modello da creare in Simulink*

Attraverso Simulink si è riuscito a realizzare il seguente schema a blocchi a retroazione unitaria

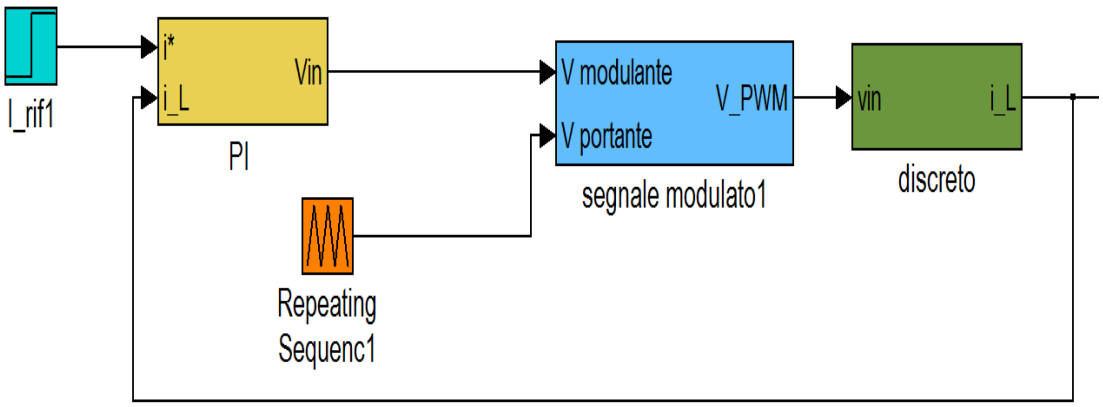


***Figura 44**-Modello retroazione unitaria*

Il PI lo si ricava dalla trasformata del sistema da controllare, che risulta nient'altro che essere un'equazione del filtro del primo ordine:

$$P(s) = \frac{1}{L\,s + R};$$

dalla quale si ricava $M = \frac{1}{|P(j\,w_t)|}$; e $\varphi = m_f - (\angle(P(s) - 180)$;

con le quali si riesce a ricavare $K_P = \cos(\varphi)\,M;$ $K_I = -sen(\varphi)\,M\,w_t;$

Ottenendo come equazione del regolatore PI $C(s) = K_P + \frac{K_I}{s};$

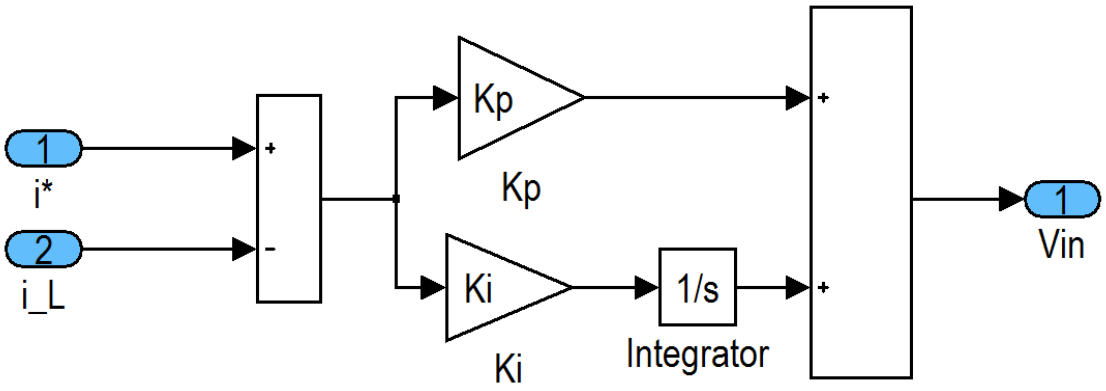


***Figura 45**-Controllore PI*

La parte dell'inverter a ponte la si è analizzata nell'esempio precedente. Va ricordato che per non aver saturazione nel PWM quando si va a confrontare il segnale modulante con quello portante (triangolare) bisogna aggiustare il segnale modulante moltiplicandolo per un guadagno molto piccolo (si è scelto come valore $2/V_{DC}$) e inoltre nella parte di discretizzazione del filtro L-R si è dovuto aggiungere un blocco ritardatore (delay) che vada a ritardare di un ciclo di clock affinché si riesca a fare tutte le operazioni logiche.

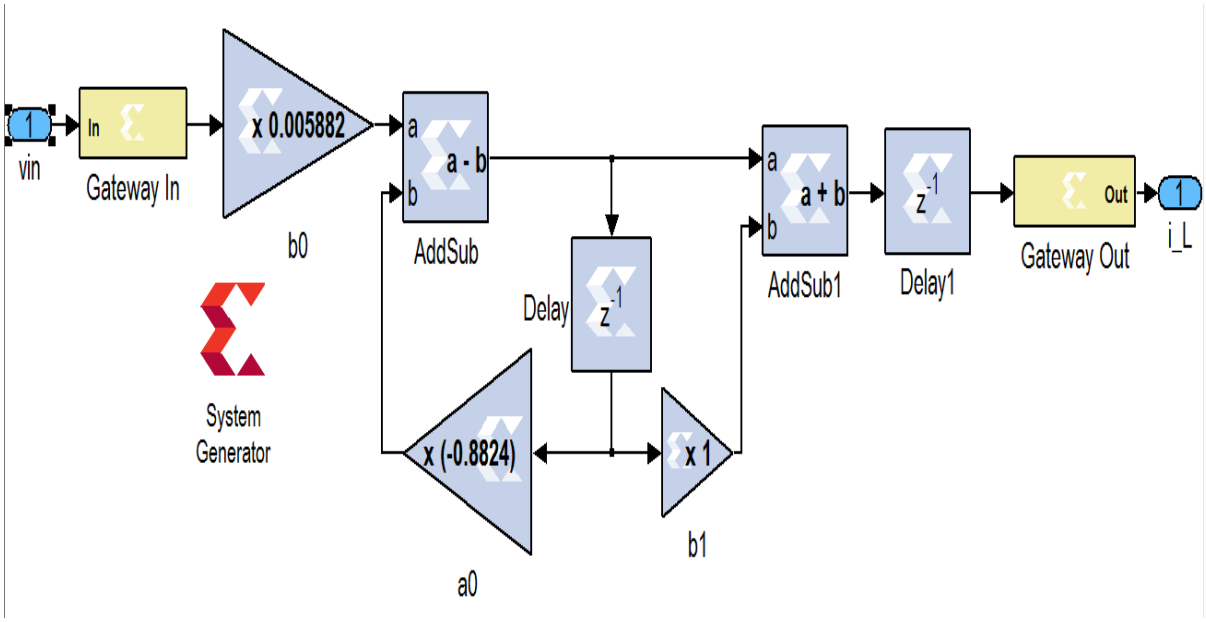


***Figura 46**-Carico L-R*

Da un segnale continuo in ingresso si è ottenuto in uscita un segnale alternato che grazie al controllore PI abbiamo accelerato la risposta del sistema

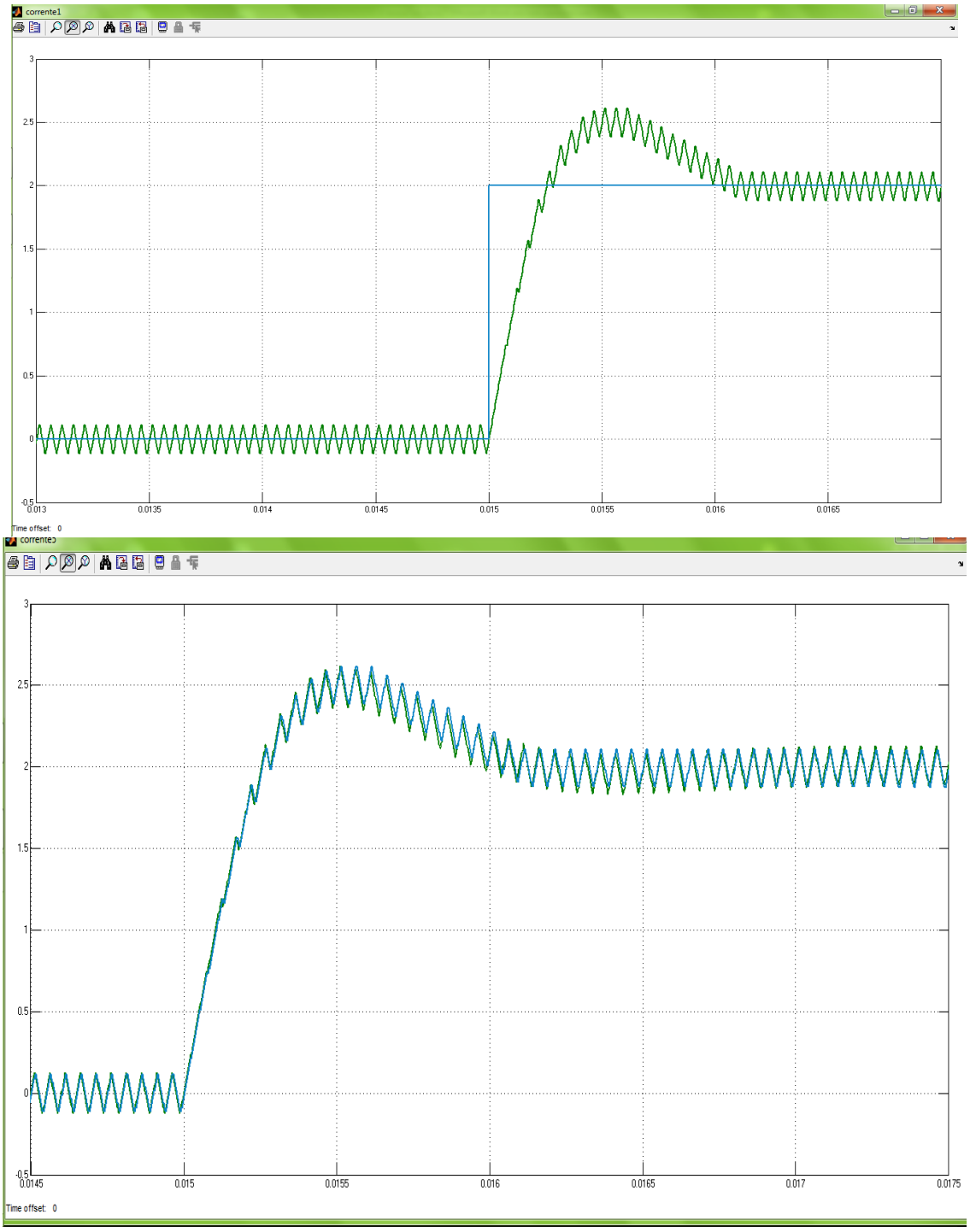


Figura 47-*Segnale in uscita discretizzato*

Conclusione

La simulazione real-time risulta una valida alternativa alla simulazione nel quale bisogna avere il prototipo già realizzato per poter testare il tutto. Il poter lavorare in un banco di prova con componenti virtuali di sicuro ha un impatto molto positivo nella fase di prototipazione, permettendo di testare in modo parallelo sia la parte hardware che software di un sistema. Si è visto come l'elettronica di potenza si sia orienta verso la simulazione real-time utilizzando varie soluzione per le piattaforme Hardware-in-the-Loop, dovuto soprattutto dalle esigenze di mercato. Si è riuscito in questo modo a ridurre la durata dei test, pur mantenendo un alto livello di sicurezza durante i test, tutto con un costo del simulazione e del test molto abbordabile sia per le aziende che per i centri di ricerca.

L'ambiente di sviluppo Xilinx System Generator, proposto nel capitolo 2, si adatta molto bene nell'essere utilizzato per la simulazione real-time e di conseguenza per Hardware-in the-Loop.

Si può vedere dagli esempi proposti come si riesce partendo dalla rappresentazione matematica degli schemi dinamici di alcuni componenti dell'elettronica di potenza, a ricavare i componenti di modellistica, indispensabili per la realizzazione di test per una sodisfacente simulazione real-time.

Tutto ciò, è agevolato dalla possibilità di utilizzare l'ambiente di sviluppo che lavora per mezzo di un modello ibridi e la possibilità di creare codici che possono essere inseriti per la fase di test nelle componenti hardware dei banchi di prova.

Anche se, non si è riusciti in pieno, con gli esempi riportati, a vedere l'effettivi vantaggi che la simulazione HIL offre con il real-time, e di conseguenza nemmeno la completa veridicità degli elementi della formula :costo-durata-sicurezza. Però ciò lo si può intravedere per mezzo dell'ultimo esempio proposto. Si potrebbe andar a testare, in un banco di prova, un controllore reale utilizzando il carico L-R modellato tramite Xilinx System Generator, caricando il suo codice in un dispositivo hardware (FPGA o DSP). Sarebbe quest'ultimo a restituire il feedback di tensione e corrente, dando il modo di fare il debug del codice senza usare potenza, cioè lavorando in condizioni non pericolose e con costi e tempi non dispendiosi.

Appendice A

In seguito vengono riportati i codici utilizzati per i file.m di Matlab negli esempi di utilizzo:

```
%% FILTRO PASSA BASSSO %%
R = 1e4;                ... Ohm
C = 1e-6;               ... F
tau = R*C;              ... costante

%calcolo della funzione di trasferimento(FdT) a tempo continuo
s = tf('s');
W = 1/(1+tau*s);

figure()
bode(W), grid;          ...  grafico la Fdt

%calcolo la funz di trasferimento nella trasformata Zeta(uso Tustin)
B_3 = 1/(2*pi*tau);     ... banda passante
fc = B_3*100;           ... frequenza di campionamento
T = 1/fc;               ... periodo di campionamento
omega = 2*pi/T;         ... pulsazione di campionamento
fn = fc/2;              ... frequenza di Nyquist
wn = 2*pi*fn;           ... pulsazione di Nyquist

%le tre costanti della FdT a tempo discreto
a0 =(T-2*tau)/(T+2*tau);
b0 =T/(T+2*tau);
b1 =1;

------------------------------------------------------------------------

%% FILTRO PASSA ALTO %%

R = 1e4;                ... Ohm
C = 1e-6;               ... F
tau = R*C;              ... costante

%calcolo della funzione di trasferimento a tempo continuo
s = tf('s');
W = (tau*s)/(1+tau*s);  ...FdT continua
figure()
bode(W), grid;

%calcolo la FdT nella trasformata Zeta(uso Tustin)
B_3 = 1/(2*pi*tau);     ... banda passante
fc = B_3*100;           ... frequenza di camp.
T = 1/fc;               ... periodo di camp.
omega = 2*pi/T;         ... pulsazione di camp.
fn = fc/2;              ... frequenza di Nyquist
wn = 2*pi*fn;           ... pulsazione di Nyquist

%le tre costanti della FdT a tempo discreto
a0 =(T-2*tau)/(T+2*tau);
b0 =(2*tau)/(T+2*tau);
b1 =-1;
```

```
------------------------------------------------------------------------
%% INVERTER %%
fs = 500;                ... Hz frequenza fondamentale;
ws = 2*pi*fs;            ... [rad/sec] pulsazione fondamentale;
fws =20*10^3;            ... frequenza di commutazione;
Tws = 1/fws;             ... tempo di commutazione;
Tm = Tws/2;
L = 2e-4;                ... induttanza
R =10;                   ... resistenza
Vdc = 1000;
tau = L/R;

s = tf('s');
P = 1/(R+L*s);


% Valori per la discretizzazione
T =Tws/20;

b0 = (T)/(T*R+2*L);
a0 = (T*R-2*L)/(T*R+2*L);
b1 = 1;

------------------------------------------------------------------------


%% CONTROLLO IN RETROAZIONE DI UN INVERTER A PONTE %%

% costruiamo il circuito RL
fs = 500;                ... Hz frequenza fondamentale;
ws = 2*pi*fs;            ... [rad/sec] pulsazione fondamentale;
fws =20*10^3;            ... frequenza di commutazione;
Tws = 1/fws;             ... tempo di commutazione;
L = 50e-3;               ... resistenza
R =0.11;                 ... induttanza
Vdc = 1000;
tau = L/R;

% Valori per la discretizzazione
T =Tws/20;
Tm =Tws/2;

wt =2*pi*fws/20;         ... banda passante
mf =60;                  ... margine di fase

% costruiamo P(s)
s = tf('s');
P = 1/(R+L*s);
[modulo,phase] = bode(P,wt);

% costuiamo il controllore C(s)
M =1/modulo;             ...
theta = mf-(phase-180); ...
theta =theta/180*pi;

Kp = cos(theta)*M;       ...
Ki = -sin(theta)*M*wt;   ...
```

```
C = Kp +Ki/s;            ... controllore
G =P*C
W = G/(1+G)


Wdisc = T*(z+1)/(z*(2*L+T*R)+(-2*L+T*R));
figure()
bode (Wdisc)

b0 = T/(T*R+2*L);
a0 = (T*R-2*L)/(T*R+2*L);
b1 = 1;
```

Bibliografia

[1] Xilinx, “Sustem Generator for DPS – Getting Started Guide”, UG639 (v11.4) dicember 2 ,2009.

[1] Christian Graf, Jurgen Maas, Thomas Schulte and Weise-Emden, “Real-time HIL-Simulation of Power Electronics” University of Applied Sciences

[3] Xilinx, “Sustem Generator for DPS – User Guide”, Release 10.1 march, 2008.

[4] M. Guarnieri, A.Stella, “Principi ed Applicazioni di Elettrotecnica”, VOLUME PRIMO, terza edizione, Edizione Progetto Padova, 2004.

[5] M. Guarnieri, A.Stella, “Principi ed Applicazioni di Elettrotecnica”, VOLUME SECONDO, terza edizione, Edizione Progetto Padova, 2004.

[6] G. Ricci, M. E. Valcher, “Segnali e Sistemi”, quarta edizione, Edizione Libreria Progetto Padova, ottobre 2010.

[7] M. Ziliotto, “Convertitori statici CC-CA”, *Fondamenti di macchine ed azionamenti elettrici*, dispense.

[8] “Inverter Trifase: Modulazione Vettoriale” Tesi di Laurea in ingegneria Elettrica, Università degli Studi di Padova, anno accademico 2010-2011.

[9] R. Oboe, “Sistemi di controllo a Segnali Campionati”, *Controlli Automatici*, dispense.

[10] www.typhoon-hil.com - www.rtds.com - www.opal-rt.com - www.ni.com

[11] Hardware-in-the-Loop Simulation, Telemark University College Department of Electrical Engineering, Information Technology and Cybernetics