Tesi Magistrale in Ingegneria Elettrica – A.A. 2019/2020

Assembly e Testing di un convertitore di potenza

Silvana Benanti
* * * * * *

Relatori
Prof. Radu Bojoi, Relatore
Fabio Mandrile, Co-Relatore

Politecnico di Torino
7 Ottobre, 2020

Sommario

Indice delle Tabelle

Indice delle Figure

Capitolo 1

Introduzione

L'energia elettrica può essere utilizzata in varie forme a seconda dell'applicazione: in corrente continua o alternata monofase o trifase a diverse frequenze. Per passare da una forma all'altra è necessario poter disporre di un sistema di conversione dell'energia elettrica, che fino ai primi anni 60' si basava quasi esclusivamente su principi elettromeccanici. I sistemi elettromeccanici convertivano prima l'energia elettrica in forma meccanica e poi da meccanica ad elettrica con caratteristiche differenti da quella originaria. Il processo storico di ottimizzazione energetica, recentemente definito anche *efficientamento* ha portato alla nascita di convertitori statici di potenza elettrica, con i quali è possibile convertire l'energia elettrica senza passare attraverso la forma meccanica, aumentando così notevolmente il rendimento. La natura di questi secondi convertitori non prevede parti in movimento, cosa che conferisce loro anche una maggiore affidabilità, oltre all'appellativo convertitori statici. Siccome poi i convertitori statici fanno uso di componenti elettronici di potenza a semiconduttore, quali diodi, transistor e tiristori, essi sono anche detti convertitori elettronici di potenza.

La classificazione dei convertitori elettronici di potenza viene fatta in funzione del tipo della sorgente elettrica e del carico:

- Convertitori AC/DC (raddrizzatori) da sorgenti in alternata a carichi in continua;
- Convertitori DC/DC (chopper) da sorgenti in continua a carichi in continua, tipicamente aventi valori di tensione diversi;
- Convertitori DC/AC (inverter) da sorgenti in continua a carichi in alternata;
- Convertitori AC/AC da sorgenti in alternata a carichi in alternata.

L'obiettivo di questa è tesi è quello di realizzare un banco prova per il testing di due inverter a IGBT controllati grazie ad un sistema che impiega una scheda

FPGA che si interfaccia con la piattaforma dSPACE, nella quale viene implementato il controllo ad alto livello, ed un microcontrollore con funzioni accessorie e di supervisione.

Le fasi del processo che verrà descritto nel corso di tutta la tesi risultano:

- Scrittura firmware per FPGA;
- Design e simulazione del controllo su PLECS;
- Scrittura codice di alto livello per dSPACE;
- Testing preliminare delle schede di controllo e di misura;
- Assemblaggio meccanico delle parti del sistema;
- Collaudo del sistema con una prova base.

Tali fasi rappresentano quindi anche gli obiettivi della presente tesi.

1.1 Il Power Stack

Gli inverter utilizzati per svolgimento di questa attività di tesi fanno parte di due esemplari gemelli di power stack della SEMIKRON.

Un power stack è l'insieme di:

- Moduli di potenza a semiconduttore;
- DC-link;
- Sistema di raffreddamento;
- Gate driver;
- Protezioni;

Si presentano di seguito i componenti dello stack utilizzato e se ne si riportano i riferimenti ai datasheet in bibliografia.

L'IGBT Module Stack della SEMIKRON è [11] mostrato in Figura 1 contiene, oltre al modulo di potenza, anche il DC-link ed il sistema di raffreddamento.

Il DC-link è composto da sei condensatori elettrolitici, di capacità equivalente pari a $7\,mF$, mentre il sistema di raffreddamento risulta essere un ventilatore centrifugo collegato ad un dissipatore alettato posto sul fondo del modulo. Questi due rappresentano la maggior parte, in termini di peso e volume, dello stack.

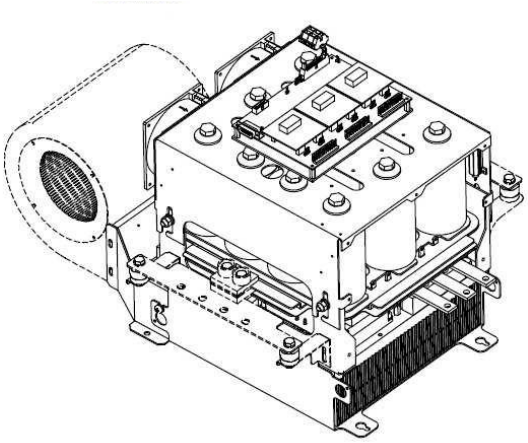


Figura 1 - IGBT Module Stack, Semikron [11].

La parte di potenza propriamente detta è costituita da un ponte trifase a diodi e da un ponte trifase a IGBT con tensione e corrente nominali rispettivamente pari a $V_{rated} = 400\,V_{ac}$ e $I_{rated} = 180 A_{rms}$, il cui schema elettrico è mostrato in Figura 2.

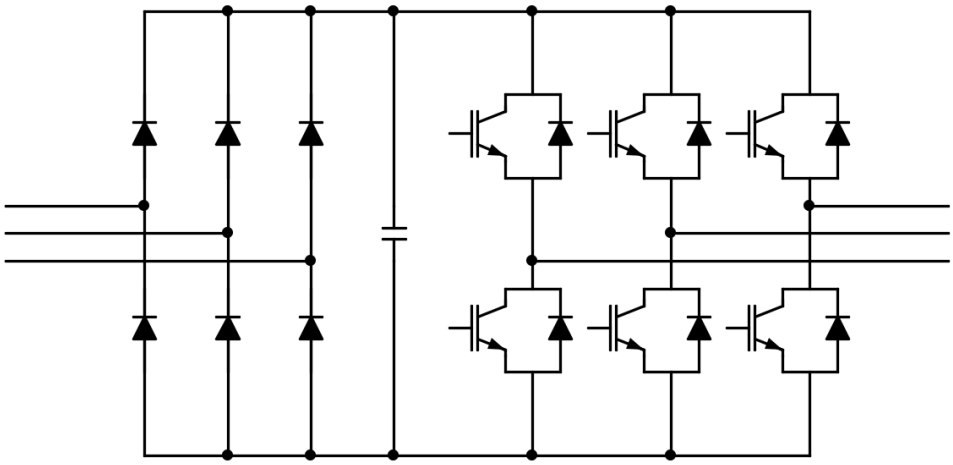


Figura 2 - Schema elettrico del modulo a IGBT.

Per quanto riguarda i gate driver e le protezioni, questi sono contenuti nelle schede mostrate nella Figura 3, di cui si riporta il riferimento al datasheet in bibliografia [12], [14]. Si nota che tali schede sono specifiche per l'hardware che pilotano.

Figura 3 - IGBT Driver Core (a sinistra) e Driver Board (a destra).

La scheda Driver Core contiene i gate driver, che sono circuiti di bassa tensione e piccola potenza che servono a pilotare i transistor di potenza, facendoli commutare; costituiscono cioè l'interfaccia tra i transistor e il circuito di controllo.

Sulla scheda Driver Board sono invece presenti i segnali relativi alle protezioni.

2.1.1 Configurazioni moduli di potenza

Poiché sono disponibili due esemplari gemelli, è possibile realizzare un sistema back-to-back. Per fare ciò si elimina il ponte a diodi da entrambi gli stack e si collegano i due DC-link, come mostrato in Figura 4.

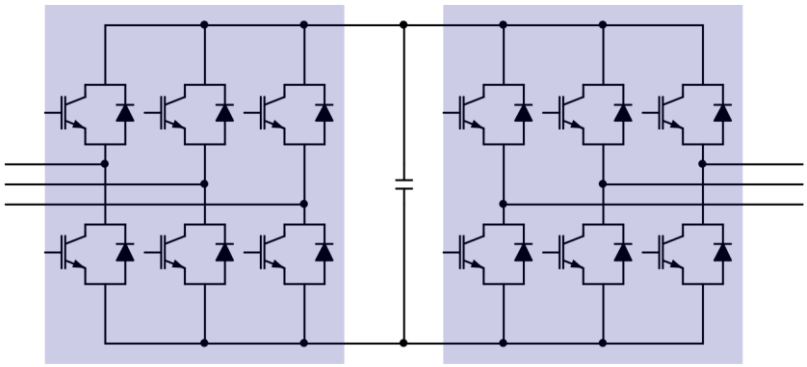


Figura 4 - Configurazione back-to back.

Il termine back-to-back indica due convertitori accoppiati in DC che interfacciano due sistemi AC qualunque, siano essi carichi o sorgenti. Poiché i due convertitori sono attivi e quindi bidirezionali in potenza questi possono, infatti, comportarsi da raddrizzatori o da inverter a seconda della direzione del flusso di potenza. Questa configurazione è utile per sistemi rigenerativi, in cui il flusso potenza può cambiare direzione.

Nel caso in cui a valle del sistema vi sia una rete trifase di alimentazione si parla di configurazione AFE + Inverter, mostrata in Figura 5, dove con l'acronimo si indica appunto l'Active Front End. In questo caso il flusso di potenza va dalla

rete verso il carico, quindi il convertitore di sinistra si comporta da raddrizzatore attivo, mentre quello di destra assume il compito di inverter.

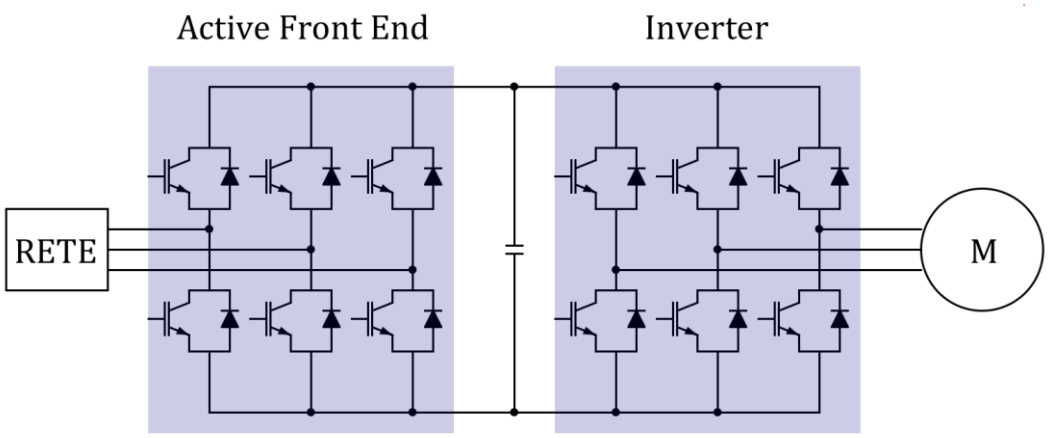


Figura 5 - Configurazione AFE + Inverter.

Un'altra possibile configurazione è che presenta due inverter alimentati da una sorgente DC esterna. Un'applicazione molto utile è, ad esempio, quella del testing a piena potenza di macchine di grossa taglia, usando un'alimentazione a potenza ridotta.

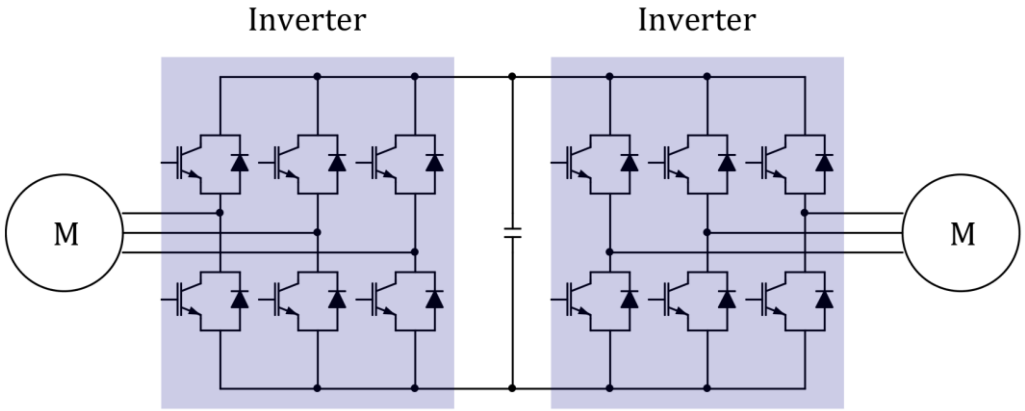


Figura 6 - Back-to-back da DC esterno.

Sono numerosissime le possibili configurazioni ottenibili con due inverter. Oltre alla back-to-back, è possibile realizzare una configurazione esafase, grazie alla quale si possono, ad esempio, controllare macchine a sei fasi o multi-trifase.

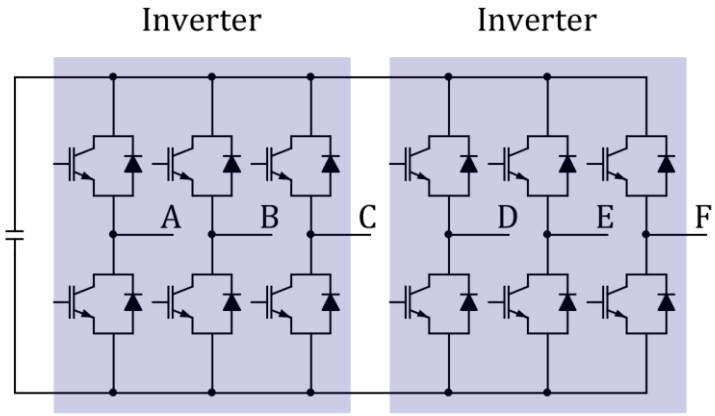


Figura 7 - Configurazione esafase.

1.2 Architettura del sistema

Per rendere il sistema operativo è necessario disporre di schede di misura e schede di controllo. Quelle utilizzate per la presente attività sono state appositamente progettate per l'interfaccia con i due inverter SEMIKRON, con l'obiettivo di sviluppare un sistema di inverter utilizzabile in varie configurazioni e interfacciabile con il software dSPACE o con un microcontrollore. Di seguito se ne presentano brevemente le funzionalità e si rimanda al documento [7] presente in bibliografia per maggiori dettagli.

La parte di controllo ha una scheda principale FPGA collegata a dSPACE, da cui partono delle ramificazioni simmetriche (una per power stack) che portano alle schede di misura di tensione e corrente e ad un'interfaccia inverter, come si mostra in Figura 8.

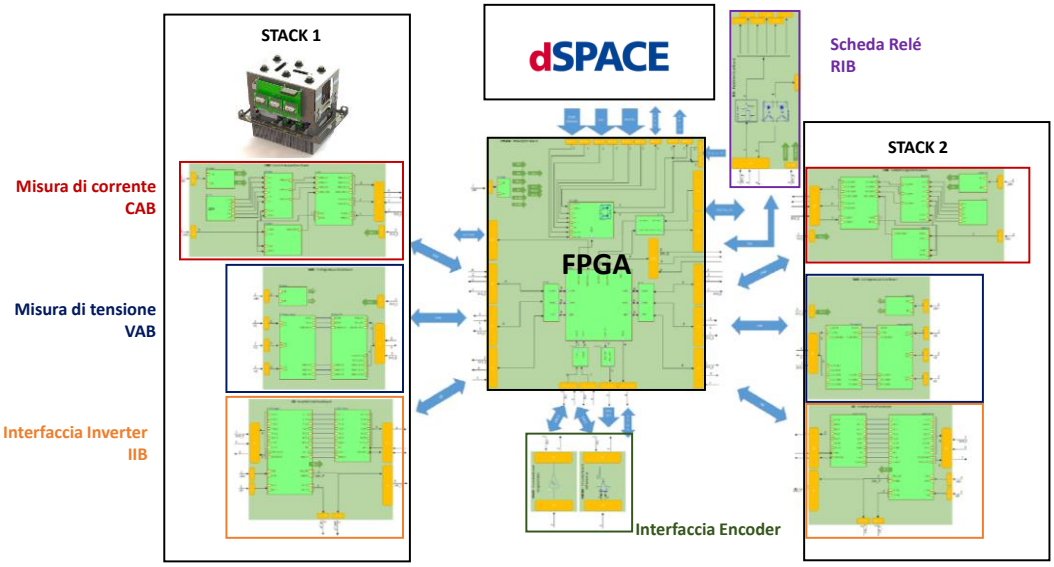


Figura 8 - Architettura del sistema.

La scheda FPGA è quindi un'interfaccia per il controllo con le schede di misura dei due stack. Si trova anche la connessione verso la scheda di controllo relè, che è invece unica per i due inverter.

La IIB (Inverter Interface Board) contiene l'adattamento delle alimentazioni e dei circuiti logici degli stati proibiti. Riceve i segnali di gamba dalla scheda di controllo e li "converte" in segnali adatti all'inverter con cui si interfaccia direttamente.

La scheda CAB (Current Acquisition Board) misura le correnti i_{abc} di fase e la tensione V_{DC} al DC-link. Le tre misure di corrente vengono fatte tramite sensori LEM, montati sulla scheda stessa. Queste, insieme alla tensione, vengono convertite tramite ADC e inviate alla scheda di controllo sotto forma di segnale seriale SPI.

La scheda VAB (Voltage Acquisition Board) è pensata per misurare le tensioni trifase ai morsetti in riferimento al punto medio dell'inverter oppure come tensioni concatenate. Le misure di tensione vengono quindi scalate, convertite in digitale in formato seriale SPI ed inviate all'FPGA.

La RIB (Relay Interface Board) contiene i sei relè necessari a gestire le configurazioni dei due inverter.

Sono poi presenti due schede di interfaccia per encoder con la possibilità di montare due diverse schede in funzione dell'utilizzo di un encoder differenziale o single ended.

Le schede di interfaccia relè ed encoder non verranno utilizzate per lo svolgimento di questa attività, in quanto non necessarie al banco test previsto.

La Figura 9 mostra lo schema di acquisizione delle grandezze di interesse. Queste vengono misurate e campionate dall'ADC, di cui si riporta al riferimento al datasheet in bibliografia [15], che converte quindi la misura in un segnale seriale di tipo SPI, il quale viene inviato in forma digitale isolata direttamente alla scheda FPGA. Tale scheda comunica con dSPACE, che riceve l'informazione, la elabora e rimanda all'FPGA i dati aggiornati utili per il controllo.

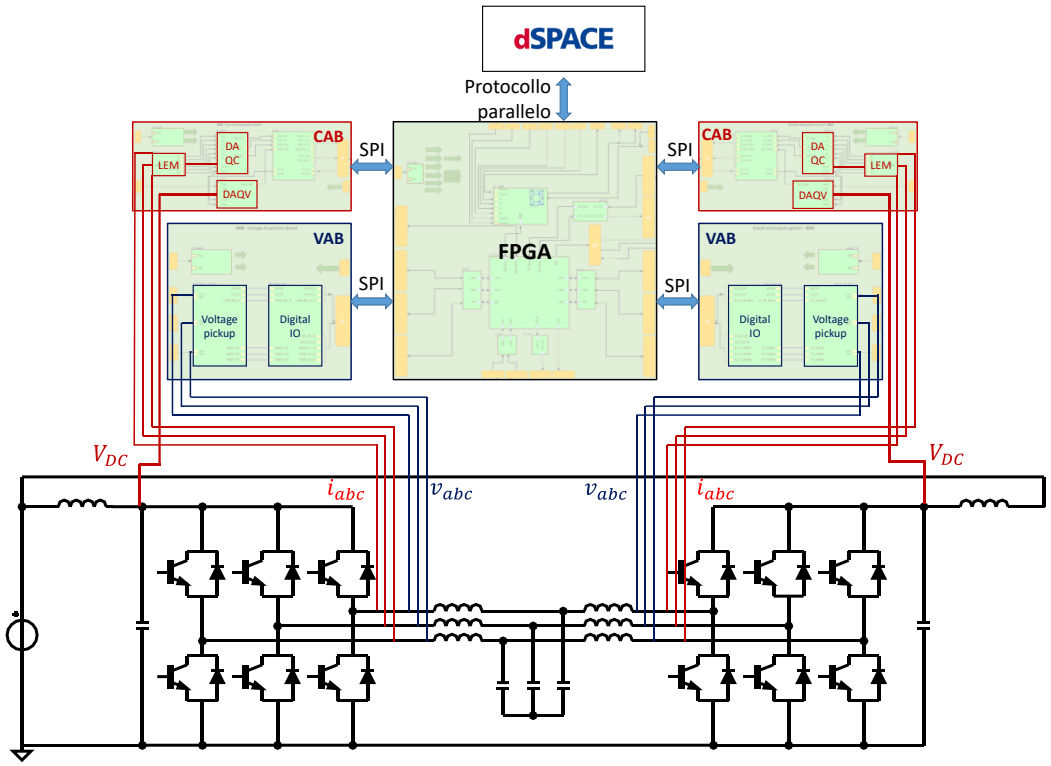


Figura 9 - Schema di acquisizione delle grandezze di interesse.

1.3 Architettura del controllo

L'architettura del controllo riguarda i compiti delle singole parti e come queste comunicano fra loro. Il controllo è composto da una scheda FPGA centrale che comunica con un microcontrollore tramite un protocollo seriale di tipo SPI e con dSPACE tramite un protocollo parallelo, come mostrato in Figura 10.

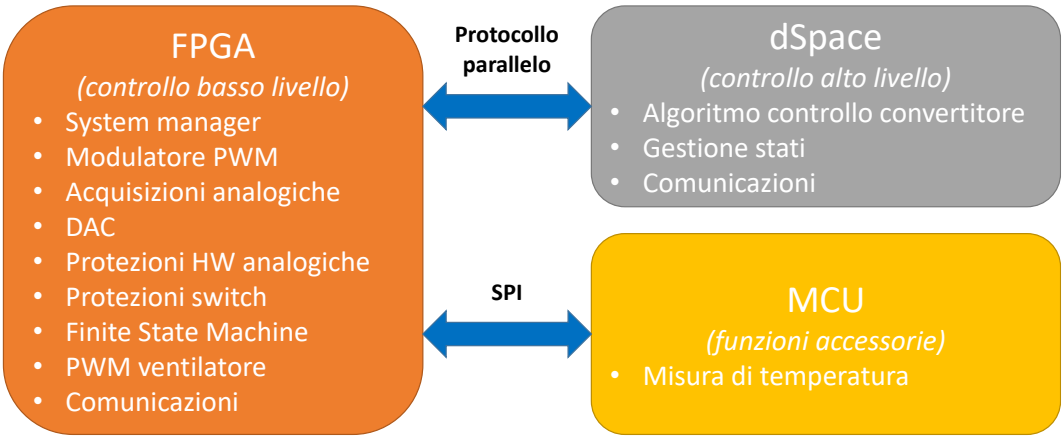


Figura 10 - Architettura del controllo.

Sulla scheda FPGA viene implementato il controllo di basso livello, che comprende il centro di temporizzazione del sistema costituito dal System Manager, il modulatore PWM per la generazione dei segnali di gamba, la gestione delle acquisizioni analogiche di tensione e corrente, un DAC con funzione di debug, le protezioni hardware analogiche, le protezioni per gli IGBT, una macchina a stati e i moduli per la gestione della comunicazione con le altre parti.

dSPACE è una piattaforma per lo sviluppo di controllori digitali multi-variabili ad alta velocità e per simulazioni real-time, permettendo di provare strategie di controllo in modo rapido, senza sviluppare una scheda dedicata per il test. Su questa viene implementato il controllo ad alto livello, in questo caso costituito da un controllo di corrente con PI.

Il microcontrollore, infine, presenta funzioni accessorie, quali la gestione dell'alimentazione del ventilatore e le misura di temperatura.

Capitolo 2

FPGA Design Workflow

I dispositivi programmabili giocano un ruolo chiave nella progettazione dei sistemi digitali. Si tratta di chip general purpose che possono essere configurati per un'ampia gamma di applicazioni. Tra questi figurano gli FPGA (Field Programmable Gate Array) introdotti per la prima volta nel 1985 dalla Xilinx.

Un FPGA è composto da un array bidimensionale di blocchi logici che possono essere connessi tramite una rete di interconnessioni programmate dall'utente.

Gli FPGA utilizzano il linguaggio di programmazione standardizzato VHDL che contiene costrutti complessi con i quali è possibile scrivere il codice di descrizione di circuiti comunque composti. Questo consente inoltre la progettazione tramite librerie e la creazione di librerie stesse definite dall'utente. Tale linguaggio consente infine, oltre alla progettazione, la simulazione del progetto.

Per lo sviluppo del codice di programmazione dell'FPGA si utilizza la piattaforma di sviluppo software rilasciata da Xilinx, cioè Vivado.

In questo capitolo si affronterà il tema della creazione di un progetto per FPGA su Vivado, spiegandone i passi chiave, e si mostrerà il progetto implementato per un sistema composto dagli elementi già presentati in Figura 10.

2.2 Creazione di un progetto Vivado

Il prima passo nella creazione di un nuovo progetto è la selezione del linguaggio desiderato (VHDL), seguito dalla scelta dell'FPGA, che in questo caso è il XC7Z010, di cui si riporta il riferimento al datasheet in bibliografia [17].

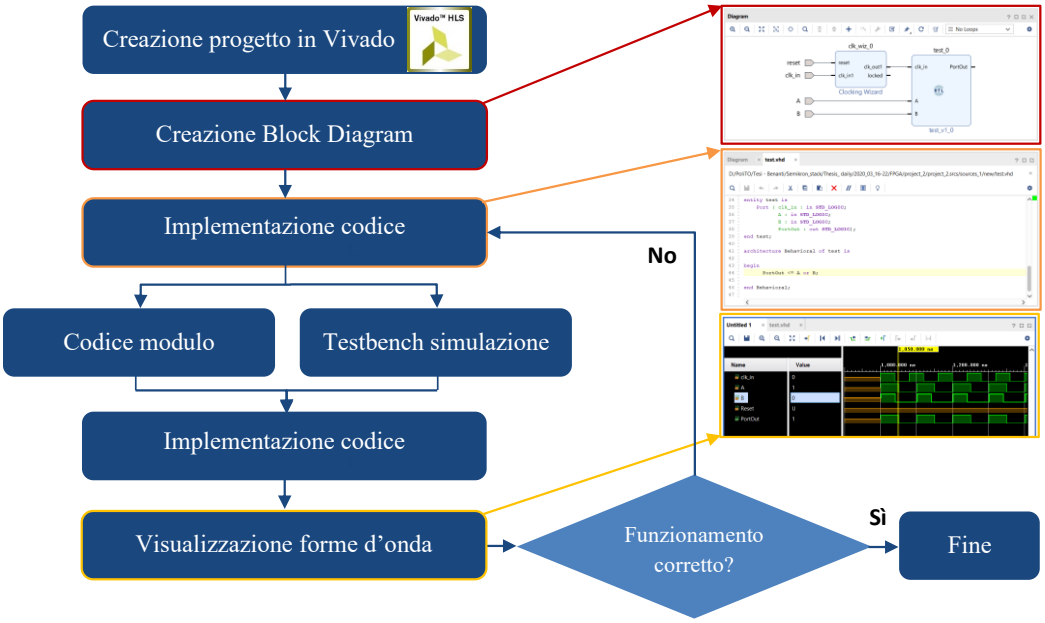


Figura 11 - Design Workflow in Vivado.

Con riferimento alla Figura 11 si osservi il flusso di lavoro da seguire nella creazione di un progetto in Vivado, del quale vengono di seguito descritti in dettaglio i passi principali. Una volta selezionato l'FPGA di interesse, si definisce il Block Diagram, nel quale si crea, modulo dopo modulo, il diagramma a blocchi del sistema che si intende testare. Il progetto può quindi essere arricchito con codici di ulteriori moduli generati dall'utente e testbench di simulazione che permettono un testing più agevole del codice prima che questo venga caricato sulla scheda. Infine, si lancia la simulazione così da verificare, tramite l'osservazione delle forme d'onda, il corretto funzionamento del sistema.

2.2.1 Il Block Diagram

A questo punto il progetto è inizializzato ed è possibile cominciare a generare il codice di livello superiore.

Ciò viene generalmente fatto come Block Design, che è un'espressione grafica di codice e che permettere di creare blocchi dotati di porte, tramite le quali i moduli si interfacciano tra loro e col mondo esterno. Una volta che il Block Diagram è stato realizzato è possibile generare automaticamente il codice VHDL per l'FPGA facendo click su "Generate Block Design".

Si nota che è possibile utilizzare i blocchi predefiniti nella libreria IP fornita da Xilinx, come ad esempio il blocco Clocking Wizard, oppure crearne di propri. È infatti possibile aggiungere dei blocchi codice, creando una nuova *design source* che ne contenga il codice, ed inserirli nel *Diagram* con la funzione "Add Module to Block Design".

A seguire un esempio di Block Diagram costituito da un Clocking Wizard ed un blocco Test generato dall'utente.

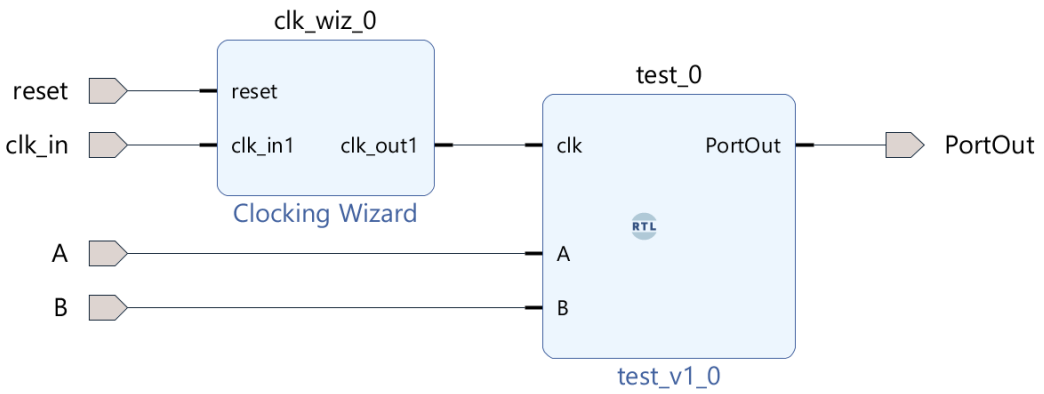


Figura 12 - Esempio di Block Design.

2.2.2 L'implementazione del codice

Una volta creato il file *design source* si deve quindi scrivere il codice del blocco che si desidera generare. Il codice è diviso in due sezioni: entity declaration e architecture body. Un entity è un'astrazione di un blocco che rappresenta un sistema completo, una scheda, un chip, una funzione o una porta logica. Una dichiarazione di entity descrive l'I/O di un progetto e può includere anche parametri utilizzati per customizzare l'entity stessa. L'architettura descrive le funzioni di un'entity.

A seguire un esempio di codice per la creazione di un blocco OR.

```
entity test is
    Port ( clk : in STD_LOGIC;
           A   : in STD_LOGIC;
           B   : in STD_LOGIC;
           PortOut : out STD_LOGIC);
end test;

architecture Behavioral of test is

begin
      PortOut <= A or B;

end Behavioral;
```

Prima di avviare la simulazione, che è possibile svolgere manualmente, risulta conveniente scrivere un testbench, ovvero un codice in VHDL che generi gli stimoli (input) per il sistema. Si crea quindi un file *simulation source* che emula il comportamento degli ingressi fisici del sistema.

A seguire un esempio di codice per la generazione di un clock in ingresso.

```
 process begin

    clk_in <= '0';
  wait for clk_in_period/2;
    clk_in <= '1';
  wait for clk_in_period/2;

end process;
```

2.2.3 La simulazione

Grazie ad alcuni tool di Vivado è possibile testare i codici sviluppati in simulazione. È dunque possibile simulare il sistema nel dominio del tempo ed osservare gli stati logici facendo click su "Run Behavioral Simulation", così da avviare la simulazione funzionale del codice a livello ideale.

A seguire un esempio di simulazione del sistema descritto.

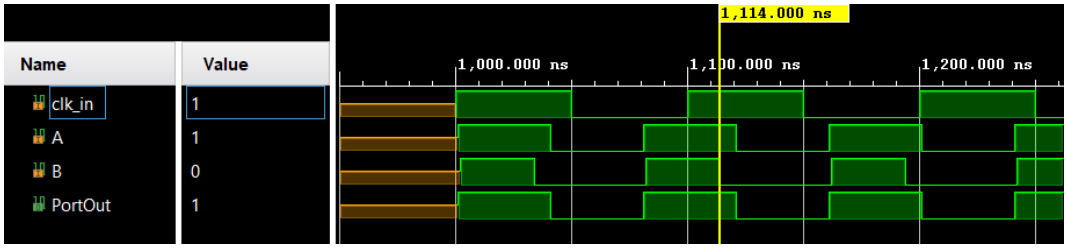


Figura 13 - Esempio di una simulazione behavioral in Vivado.

2.3 Implementazione del sistema da testare

Il sistema da testare è composto da un generatore di clock, un system manager, un modulatore PWM ed un modulo per la gestione degli ADC. Il sistema viene poi completato con le opportune protezioni hardware analogiche protezioni IGBT, un DAC con funzione di debug, una macchina a stati ed un modulo per la comunicazione con dSPACE. Il sistema completo viene riportato nella Figura 62.

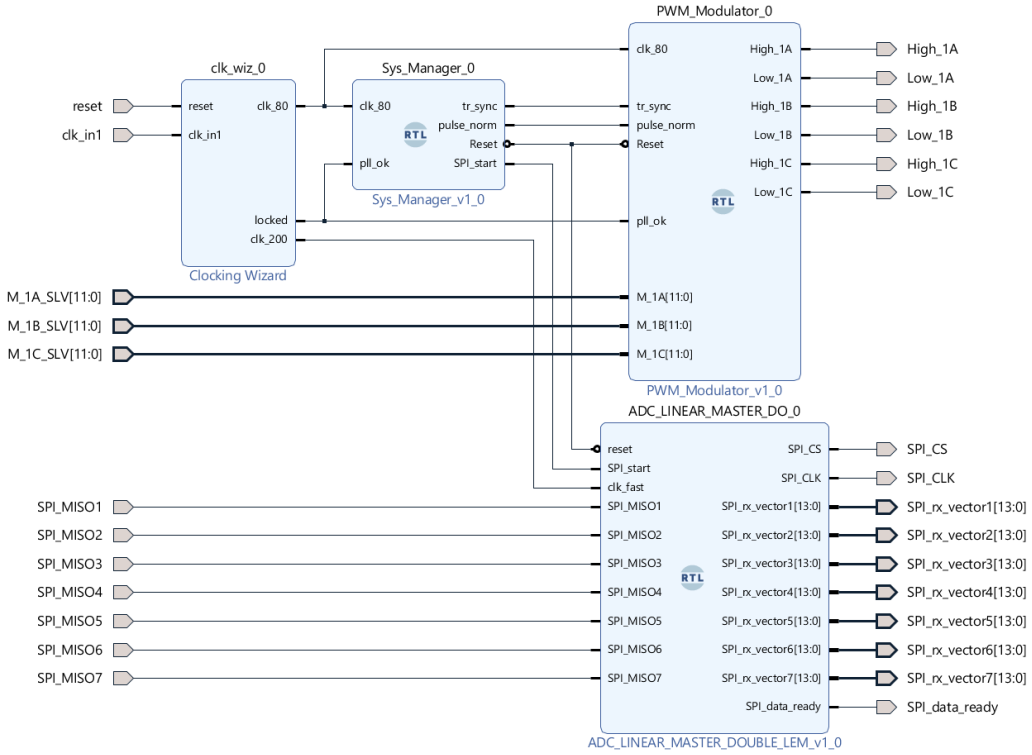


Figura 14 - Sistema da testare.

I moduli, presentati nel loro insieme e con le opportune connessioni, vengono di seguito descritti singolarmente.

2.3.1 Clocking Wizard

Come generatore di clock si utilizza il blocco Clocking Wizard prima citato, che è un blocco preesistente nella libreria IP di Vivado e che a partire da un clock ne genera altri a diversa frequenza. Questo aiuta a creare il circuito di clock per la frequenza di clock, la fase e il duty cycle richiesti utilizzando blocchi di segnali misti progettati per supportare il disallineamento del clock della rete, la sintesi di frequenza e la riduzione delle irregolarità (jitter). Tali blocchi sono il mixed-mode clock manager (MMCM) o il phase-locked loop (PLL), di cui il primo supporta più funzionalità, come è possibile verificare sull'apposito datasheet delle risorse di clock degli FPGA serie-7 [18]. Il Clocking Wizard aiuta infine a verificare la frequenza di clock generata in uscita nella simulazione, fornendo un design di esempio sintetizzabile che può essere testato sull'hardware.

Si entra quindi nel sistema con un clock a 100 MHz, che emula il sistema fisico e che sarà quindi una porta nel *Diagram*.

Figura 15 - Blocco di generazione del clock di ingresso.

Nel testbench si genera un process che produce tale clock a 100 MHz con $duty = 0{,}5$, come mostrato precedentemente. Si settano come output:

- Un clock a 80 MHz, che è quello necessario al modulatore PWM considerati i limiti di $Nbit$ e f_{sw}, che verranno successivamente discussi;
- Un clock a 200 MHz, che sarà il clock veloce di temporizzazione dei processi.

Si abilita inoltre l'uscita *locked*, che indica quando il MMCM/PLL ha raggiunto l'allineamento di fase all'interno di una finestra predefinita e la corrispondenza della frequenza all'interno di un intervallo predefinito. Il segnale di *locked* verrà disinserito se il clock di ingresso si arresta o viene violato l'allineamento di fase, ad esempio, con uno spostamento di fase del clock di ingresso.

Si verificano quindi i risultati in simulazione.

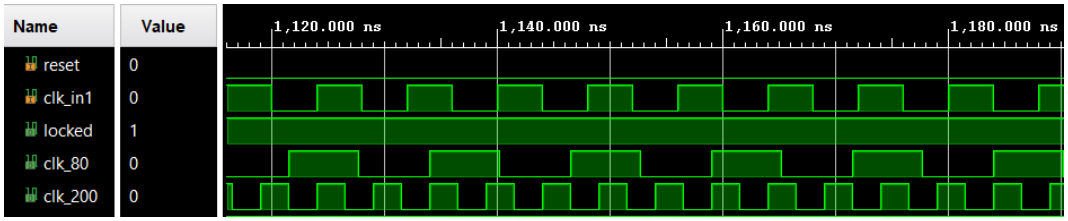


Figura 16 - Simulazione Clocking Wizard.

2.3.2 System Manager

Il System Manager è il centro di temporizzazione del sistema.

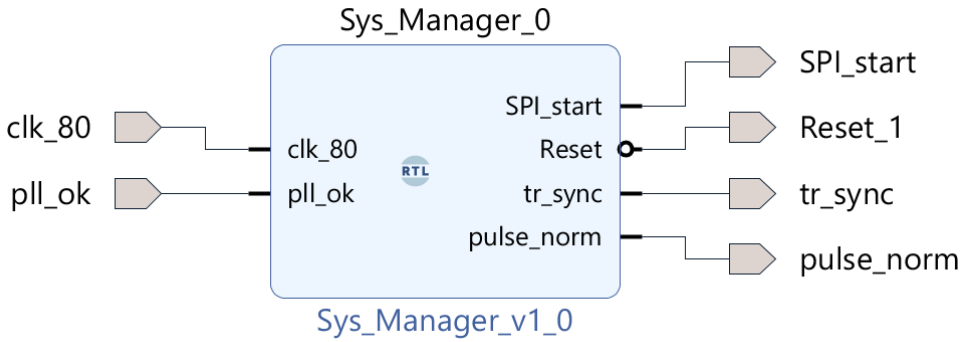


Figura 17 - System Manager.

Questo riceve in ingresso il clock a 80MHz generato dal Clocking Wizard, con il relativo segnale di stabilità dello stesso, e genera gli impulsi di sincronismo per il modulatore; gestisce inoltre lo stato di reset per i successivi moduli. Di seguito un estratto dal codice che mostra l'implementazione della procedura.

```
process(pll_ok,clk_80)
begin
if(pll_ok='0')then
 Reset<='1';
 count_pwm<=zero_T;                --timer degli eventi
else
 if(clk_80='1' and clk_80'event)then
  count_pwm<=count_pwm+uno_T;
  Reset<='0';
  if(count_pwm=value_tr_sync)then   -- syncronismo delle triangole
                                       reset
   tr_sync<='1';
  else
   tr_sync<='0';
  end if;
  if(count_pwm(N-2 downto 0)=value_norm_tr(N-2 downto 0))then --
                                       normalizzazione modulatori
   tr_norm<='1';
  else
   tr_norm<='0';
  end if;
 end if;
end if;
```

```
end process;
pulse_norm<=tr_norm;
```

Il System Manager è inoltre responsabile della temporizzazione delle acquisizioni dell'ADC sulla base della modulazione. Per fare ciò si sincronizza il campionamento delle grandezze di interesse sul vertice alto della triangola del modulatore PWM, come mostrato dal seguente codice VHDL.

```
process(START_ADC,count_pwm)
begin
    if(count_pwm=value_tr_up)then
      START_ADC <= '1';
    else
      START_ADC <= '0';
    end if;
end process;
SPI_start<=START_ADC;
```

L'Analog to Digital Converter si occupa dell'acquisizione delle tre correnti di fase, della tensione del DC-link e delle tensioni AC concatenate. Sebbene la scelta dell'istante di campionamento non rappresenti una criticità nell'acquisizione della tensione DC e della terna delle tensioni alternate, lo stesso non vale per le correnti AC di fase. La tensione al DC-link, infatti, risulta molto filtrata e la terna di tensioni può anch'essa essere efficacemente filtrata poiché si trova ad una frequenza (50 Hz) molto inferiore rispetto a quella della modulazione (10 kHz). D'altro canto, le correnti di fase sono soggette ad un forte ripple e risulta per questo opportuno campionarle a metà del tempo di ON dello switch (vertice alto della portante triangolare), così da acquisire il loro valor medio.

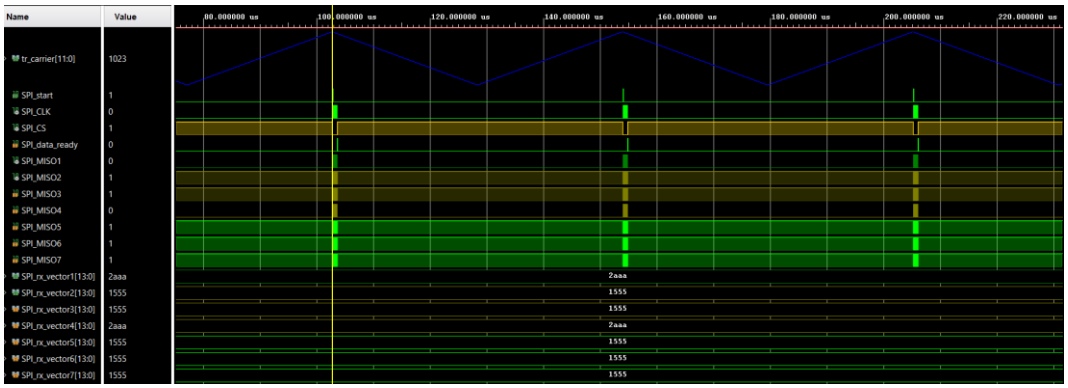


Figura 18 - Sincronizzazione dell'acquisizione rispetto alla modulazione.

2.3.3 Modulatore PWM

La Pulse Width Modulation è una tecnica che permette di controllare un'uscita digitale in modo che la tensione media di questa uscita in un dato intervallo di tempo (detto periodo di commutazione) risulti pari al valore di tensione moltiplicata per il duty-cycle d:

$$d = \frac{T_{on}}{T_{sw}}$$

Dove T_{on} è il tempo in cui l'uscita permane allo stato alto (1 logico), mentre T_{sw} è il tempo totale ($T_{on} + T_{off}$); per quanto detto quindi, se la tensione in ingresso ad un inverter è V, allora la tensione media in uscita in un periodo di switching sarà:

$$v = V \cdot d$$

Questa tecnica viene usata per il controllo di inverter, chopper, ecc. Infatti, se l'uscita dell'FPGA comanda l'accensione e lo spegnimento di un componente di potenza (es: IGBT), è possibile modulare una tensione di uscita che, mediamente, abbia una forma d'onda desiderata (per esempio una sinusoide).

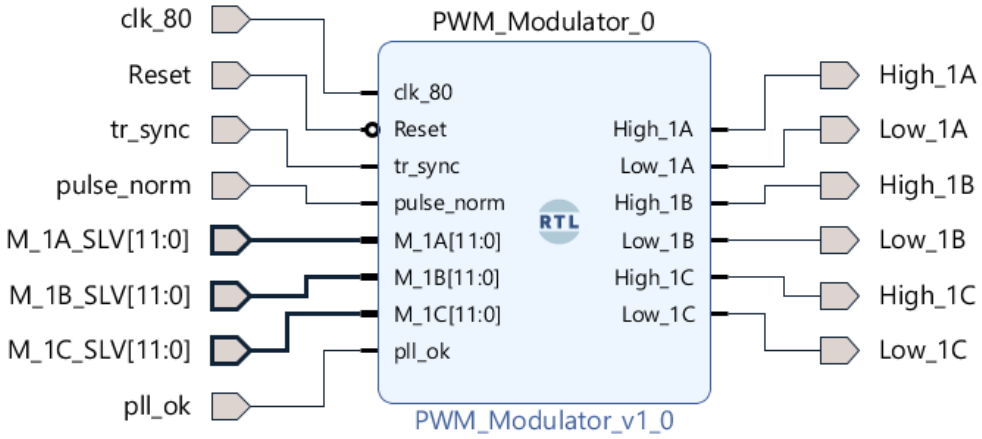


Figura 19 - Modulatore PWM.

Un modulatore PWM ha il ruolo di generare la funzione di commutazione $q(t)$ confrontando un segnale di comando $\mathrm{m}(t)$ e un segnale periodico di periodo pari al periodo di commutazione, ad esempio una triangola simmetrica, chiamato segnale portante $tr(t)$.

Gli ingressi fisici del modulatore sono:

- clk 80: clock appositamente generato dal Clocking Wizard;
- M_1A, B, C: indici di modulazione. Per questo viene creata una porta che genera il segnale sotto forma di standard_logic_vector a Nbit, il quale viene poi internamente convertito in numero signed a Nbit;
- pll_ok: corrisponde al segnale di locked del Clocking Wizard ed indica quando il clock in ingresso è stabile e può essere usato.

Il blocco riceve poi gli impulsi di sincronismo ed il segnale di reset dal System Manager.

Il blocco del modulatore PWM qui implementato è internamente composto da un blocco generatore di triangola, che è il segnale portante, e dei blocchi comparatore. Mentre c'è un solo generatore di triangola, infatti, il comparatore è instanziato 3 volte, una per ogni gamba dell'inverter. Il segnale del generatore viene quindi distribuito ai tre comparatori che confrontano l'indice di modulazione M, sottoforma di numero signed a $N\ bit$, con il segnale della triangola $tr\_carrier$, anch'esso realizzato come signed a $N\ bit$. Confrontando questi due numeri nella stessa base binaria si ottengono in uscita i comandi di gamba degli switch alti e bassi.

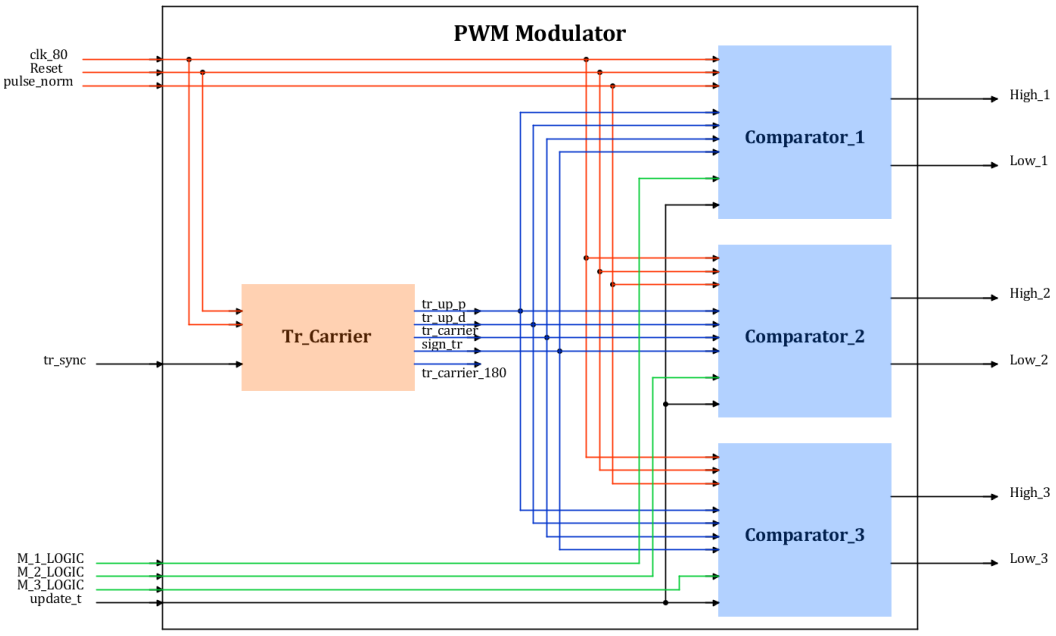


Figura 20 - Schema a blocchi del modulatore PWM.

Generatore di Triangola

Il blocco generatore di triangola è quello incaricato di fornire il segnale portante al modulatore, ovvero la triangola. Questa viene generata come segue.

```
if(Reset='0')then                          -- FUNZIONAMENTO ORDINARIO

if(slope_tr='1')then                       -- se slope positivo
    count_carrier<=count_carrier+one_T;    -- incremento della triangola
    count_carrier_N<=count_carrier_N-one_T;
    tr_down_p<='0';                         --reset impulso di underflow
    if(count_carrier=tr_overflow)then       -- al vertice positivo overflow
    slope_tr<='0';                          -- cambio pendenza
    tr_up_p<='1';                           -- se overflow
    end if;
else                                        -- se slope negativo
    count_carrier<=count_carrier-one_T;     -- decremento triangola
    count_carrier_N<=count_carrier_N+one_T;
```

```
    tr_up_p<='0';                          -- reset impulso di underflow
    if(count_carrier=tr_underflow)then     -- al vertice negativo
     slope_tr<='1';                        -- cambio pendenza
     tr_down_p<='1';                       -- set impulso di underflow
    end if;
end if;
```

Si realizza cioè una gradinata che aumenta di un'unità ad ogni colpo di clock fino al raggiungimento del suo valore massimo, momento nel quale cambia pendenza. Analogamente questa decresce fino a che il generatore non riceve il segnale di underflow e allora torna ad avere pendenza positiva.

All'interno del blocco viene inoltre attuata la sincronizzazione della portante al controllo. Quando il generatore riceve il segnale di reset, il counter della triangola viene riportato a 0, così come tutti gli impulsi interni e la pendenza viene impostata al valore scelto dall'utente tramite il parametro $slope\_ini$.

```
    else                                   -- RESET CONDITION

    count_carrier<=zero_T;                 -- set counter a zero
    count_carrier_N<=zero_T;
    slope_tr<=slope_ini;                   -- set pendenza iniziale triangola
    Reset_cmp<='1';                        -- set reset comparatori
    count_reset<="0000";                   -- reset contatore di reset cmp unit
    tr_down_p<='0';                        -- set impulso di underflow
    tr_up_p<='0';                          -- set impulso di overeflow
    end if;
```

Requisiti del clock di ingresso

Il corretto funzionamento del sistema prevede una scelta oculata del valore della frequenza del clock in ingresso f_{clk}. Questa risulta infatti legata alla frequenza di commutazione f_{sw} ed alla discretizzazione espressa in numero di bit.

Se l'intervallo fra due fronti ascendenti consecutivi è il periodo di switching T_{sw} e il periodo fra due fronti di gradini adiacenti è il periodo di clock del sistema T_{clk}, come mostrato in Figura 21, allora basta considerare che in un periodo di commutazione bisogna contare $2 \cdot 2^N$ colpi di clock. Partendo da 0, infatti, il contatore deve raggiungere il suo valore massimo, diminuire fino al minimo e tornare poi a 0 alla fine del periodo. Rappresentando tali valori col metodo del complemento a 2, ciò vuol dire passare dallo 0 a $tr_{up} = 2^{N-1} - 1$, fino a $tr_{down} = 2^{N-1}$ ed infine nuovamente allo 0. Si osserva però che, in questo caso specifico, per questioni di simmetria si è scelto di non sfruttare tutti i bit a disposizione per la generazione della triangola, ma di limitare i suoi valori massimo e minimo a $\pm 2^{N-2}$.

Con riferimento al sistema in studio, dove la frequenza di commutazione è $f_{sw} = 10kHz$ e la discretizzazione scelta è $N = 12bit$, la frequenza del clock in ingresso deve essere la seguente:

$$T_{clk} = \frac{T_{sw}}{4 \cdot 2^{N-2}} \rightarrow f_{clk} = f_{sw} \cdot 2^N = 80MHz$$

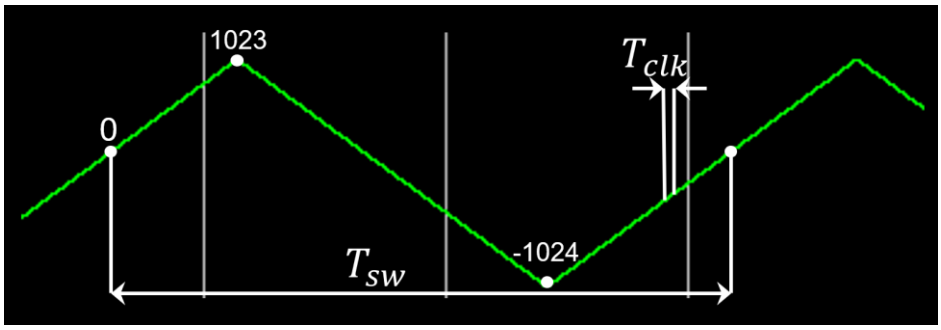


Figura 21 - Requisiti del clock in ingresso.

Comparatore

Il comparatore determina lo stato degli interruttori confrontando il segnale portante $tr(t)$ e il segnale modulante $m(t)$.

È ora interessante mettere in relazione esplicita il segnale $m(t)$ con il duty-cycle PWM risultante d. Calcoli semplici mostrano che, in ciascun periodo di modulazione, in cui si assume $m(t) = m$ costante, vale la seguente equazione:

$$\frac{m}{d \cdot T_{sw}} = \frac{tr_{pk}}{T_{sw}} \rightarrow d = \frac{m}{tr_{pk}}$$

Se si assume che il segnale modulante cambi lentamente nel tempo, rispetto al segnale portante si può ancora considerare il precedente risultato corretto.

Normalizzazione della triangola e dell'indice di modulazione

Il comparatore digitale qui implementato contiene al suo interno un'unità di normalizzazione dei segnali, necessaria se si vuole garantire la simmetria dei periodi PWM. Tale simmetria riguarda non solo la costruzione del segnale portante triangolare, ma anche l'attuazione di tempi morti fra l'apertura di uno switch e la successiva chiusura dell'altro switch della stessa gamba.

L'attribuzione di tempi morti agli indici di modulazione, questione che viene chiarita in seguito, introduce infatti un problema di simmetria. Per ora basti sapere che dagli indici di modulazione di gamba M, in ingresso al sistema, vengono costruiti altri due segnali Mp e Mn. Questi vengono rispettivamente utilizzati per il comando degli switch alti e bassi, ridotti di una certa quantità rispetto al segnale

originale per garantire un sufficiente ritardo tra l'apertura e la chiusura degli interruttori posti su una stessa gamba, che assicuri che non si verifichi mai un cortocircuito.

Per avere dei periodi simmetrici con una triangola bipolare, questa deve essere costituita da valori appartenenti ad un intorno circolare di 0. In un sistema a 12 bit questo vuol dire che il segnale può variare al massimo fra -1024 e $+1024$. Infatti, la tecnica di rappresentazione dei numeri binari in complemento a due permetterebbe un'oscillazione dissimmetrica fra -1024 e $+1023$, con solo 11 bit a disposizione.

Così come la triangola, anche gli ingressi al sistema devono avere valori compresi nell'intervallo ± 1024. In particolare, si fa riferimento agli indici di modulazione che sono nominalmente composti da 12 bit, ma di cui solo $N-2$ bit risultano utili alla comparazione. Questi infatti hanno valori effettivamente compresi nell'intervallo ± 1023, ovvero potrebbero essere descritti con 10 bit, più il segno. Poiché il sistema gira a 12 bit risulta quindi necessario che anche gli indici M vengano convertiti in valori a 12 bit, senza però alterarne la scala. Per fare ciò si attribuisce il valore assoluto dell'indice di modulazione al segnale M_abs, che viene quindi shiftato in avanti ed arricchito del segno.

```
  if((update_t(0)='1'    and    sign_tr='1')    or    (update_t(1)='1'    and
sign_tr='0'))then                          --condizione di udpate
      M_buf<= signed(M);
      start <= '1';
      count <= "0001";
     end if;
    end if;
    if(start = '1')then
     count <= count+"0001";
     case counti is
     when 1 => if(M_buf(N-1) = '1')then  -- if M<0
         segno_M <= '1';
         if(M_buf < max_n)then           -- saturation at max n
          M_abs <= max_p;
         else
          M_abs <= -(M_buf);
         end if;
        else                             -- if M>0
         segno_M <= '0';
         if(M_buf > max_p)then           -- saturation at max p
          M_abs <= max_p;
         else
          M_abs <= M_buf;
         end if;
        end if;
      when 2 => M_shift <= M_abs(N-1) & M_abs(N-3 downto 0) & '0';
      when 3 => M_24    <= M_shift*norm_M;
      when 4 =>if(segno_M = '0')then
         M_norm <= M_24(2*N-2 downto N-1);
        else
         M_norm <= -M_24(2*N-2 downto N-1);
        end if;
      when 5 => Mnforce<=MaxP2+(Maxtr-Mp); -- exit dead time zone index
        Mpforce<=MinN2+(Mintr-Mn);
      when 6 => if (Mnforce<MaxP2)then
         Mnforce<=MaxP2;
        end if;
        if (Mpforce>MinN2)then
         Mpforce<=MinN2;
        end if;                           -- STD modulation index
```

```
when 7 => Mp <= M_norm-dead_time;     -- M for switch high
   Mn <= M_norm+dead_time;            -- M for switch low

when others => start <= '0';
   count <= "0000";
end case;
```

Si noti inoltre che l'operazione di normalizzazione viene eseguita solo quando si ricevono dei nuovi indici di modulazione. Questi possono arrivare in qualunque momento all'interno del periodo PWM, per cui immediatamente prima di un vertice della triangola viene generato un impulso che fa partire la normalizzazione. Poiché in un periodo sono presenti due vertici, uno positivo ed uno negativo, bisogna generare due impulsi per periodo per ottenere il *double refresh.* Se in un contatore che va da 0 a $N-1$ si vuole riprodurre un impulso due volte al periodo, è necessario che solo i primi $N-2$ valori coincidano, generando così due impulsi simmetrici all'interno del periodo. In particolare, con riferimento al codice sopra riportato, se del valore $value\_norm\_tr$ si considerano solo i primi $N-2\,bit$ si ottiene il passaggio di questo numero all'interno del $count\_PWM$ due volte.

La costante $value\_norm\_tr$ è definita come $T-50$, cioè è pari al periodo del counter del modulo a meno di 50 colpi di clock. Per capire il motivo di tale definizione si osserva il codice della normalizzazione all'interno del comparatore precedentemente riportato, nel quale si può notare una logica sequenziale. Ogni passo corrisponde ad un colpo di clock, per un totale di 7 colpi di clock a 80 MHz.

La normalizzazione va lanciata il più vicino possibile al vertice, in corrispondenza del quale vengono attuati i nuovi comandi. Si considera però che l'operazione di moltiplicazione al passo 3 può, per alte frequenze o elevati numeri di bit, risultare onerosa e si decide quindi di lasciare un ampio margine che permetta la propagazione di eventuali ritardi.

Gestione del dead time

Il problema che si vuole affrontare in questa sezione è noto come il *dead time* di commutazione. È evidente che in nessuna condizione dovrebbe essere consentita la conduzione simultanea di entrambi gli switch. Ciò comporterebbe infatti il cosiddetto cortocircuito di gamba, portando a una circolazione di corrente incontrollata attraverso gli interruttori e, molto probabilmente, ad un danno irreparabile all'inverter. Qualsiasi modulatore, qualunque sia la sua legge di attuazione e modulazione, deve quindi essere protetto dal verificarsi di questo evento.

Nell'ipotesi di commutazione ideale, il verificarsi della conduzione contemporanea degli switch può essere facilmente prevenuto imponendo, in qualsiasi circostanza, segnali logici di gate complementari ai due interruttori; sfortunatamente, nei casi reali, questa non è una condizione sufficiente per evitare la conduzione contemporanea. Dovrebbe essere noto dalle conoscenze di base dell'elettronica di potenza che le commutazioni reali richiedono una quantità finita di tempo e che il tempo di commutazione è una funzione complessa di diverse

variabili come la corrente e la tensione di commutazione, la corrente di pilotaggio del gate, la temperatura e così via. È quindi impossibile fare affidamento su segnali di gate complementari per proteggere l'inverter. Una protezione efficace contro il fenomeno descritto viene implementata introducendo tempi morti di commutazione, vale a dire ritardi adeguati tra l'apertura di uno switch e la chiusura dell'altro.

Tale misura di protezione implica una difficoltà implementativa nella costruzione del codice di controllo degli interruttori. Risulta infatti necessario ridurre i segnali di gate di una quantità corrispondente al dead time desiderato, operando in modo simmetrico sui segnali di comando dei due switch. La difficoltà maggiore riguarda la condizione di saturazione, ovvero la condizione nella quale il segnale logico di comando ha una durata inferiore al tempo morto. Nei dintorni dei vertici positivo e negativo della triangola si definisce allora una fascia di dead time, entrando nella quale non si ha più una regolazione continua, ma si clampa il segnale a 0 o a 1, passando quindi dal valore limite al vertice senza utilizzare i valori intermedi.

Di seguito viene presentato un estratto del codice utilizzato, dove si mostra la logica implementata nelle situazioni di interesse. Per semplicità di lettura si riporta solo la parte di codice riguardante l'ingresso nella fascia di dead time e l'uscita dalla fascia di dead time relativi al vertice positivo. Analogamente si presenta la condizione di normale funzionamento solo relativa allo slope positivo della triangola.

Si osserva la scelta di utilizzare comparatori di uguaglianza, così da limitare le risorse hardware utilizzate.

```
    process(Reset,clk_80)
    begin
    if (Reset='0')then
     if(clk_80='1' and clk_80'event)then
      if(tr_up_p='1')then   -- impulso di overflow
       if(Mn>=Maxtr)then    -- Se Mn>Tr_max (non comando più lo switch low)
        Mn_c<=Mn;           -- Mnc=Mn
        if(Lbuf='1')then    -- se sul vertice positivo lo switch basso è
chiuso
         Lbuf<='0';         -- apro lo switch basso
         Mp_c<=MaxP2;       -- e imposto Mp_c pari a Tr_max-2*dead_time
        else                -- se Lbuf è già uguale zero
         Mp_c<=Mp;          -- imposto Mp_c a Mp
        end if;
       else          -- se non sono in condizione di low side sempre aperto
        Hbuf<='0';   -- apro lo switch alto
        if(Lbuf='0' and Mn<Mnforce)then  -- se lo switch basso è aperto e
il Mn è inferiore a Mnforce
         force_Lbuf<='1';  -- applico il forzamento a 1 else non faccio
niente la lascio lo switch come era al vertice
       end if;
       Mp_c<=Mp;    -- caricamento nuovi indici per il periodo in corso
       Mn_c<=Mn;    --
      end if;
```

Se il modulo non riceve alcun segnale di Reset e c'è un evento di clock, allora si passa a verificare se ci si trova o no su uno dei vertici, che rappresenta la prima criticità. A questo punto si carica l'indice di modulazione precedentemente

normalizzato nella *compare unit* per iniziare le comparazioni per il periodo PWM in corso.

Nel caso di ingresso nella fascia di dead time si passa dalla condizione di normale funzione alla condizione di saturazione, per la quale uno dei due switch non viene più comandato, ad esempio per $Mn > 0$ non si comanda più lo switch basso. Se ci si trova sulla triangola con $slope = 1$ e $Mn > Max\_tr$ allora: se lo switch basso era chiuso lo si apre immediatamente ($Lbuf = 0$) e al comparatore non si manda il dato Mp calcolato nella normalizzazione, ma si usa $MaxP2$, ovvero l'indice corretto con il dead time; se invece $Mn > Max\_tr$ ma lo switch era già aperto allora su Mp non va fatta nessuna azione particolare, ma si può chiudere lo switch positivo in un momento qualsiasi poiché il negativo è già aperto.

Per quanto riguarda l'uscita dalla fascia di dead time, questa riguarda il passaggio dalla condizione di saturazione a quella di normale funzionamento. Nella zona di confine si ha l'apertura dello switch alto (se era chiuso) e l'apertura di quello basso, evitando il cortocircuito. Si apre allora subito lo switch alto e, se lo switch basso era aperto ($Mn < Mn\_force$) allora si applica il forzamento a 1 e si lascia lo switch com'era sul vertice, altrimenti si è in condizioni normali.

Si può dunque caricare il duty cycle perché non ci si trova in zone a rischio di cortocircuiti.

Analogamente si attua sul vertice negativo.

```
     else                                -- se non sono sui vertici
      if(sign_tr='1')then                -- positive slope of the carrier
       if(force_Hbuf='1')then            -- out DEAD TIME zone se abilitato
force_Lbuf
       if(tr_carrier=Mpforce)then        -- quando la trinagola vale Mpforce
        Hbuf<='1';                       -- chiudo lo switch alto
        force_Hbuf<='0';                 -- elimino il forzamento
       end if;
      else                               -- the high side is close
       if(Mp_c=tr_carrier)then           -- if MP=tr
        Hbuf<='0';                       -- open high side switch
        Lbuf<='0';                       -- reset open low side switch
       end if;
       if(Hbuf='0')then                  -- if the high side switch is open
        if(Mn_c=tr_carrier)then          -- if Mn=tr
         Lbuf<='1';                      -- close the low side switch
        end if;
       end if;
```

Si esamina la situazione fuori dai vertici, facendo riferimento al codice sopra riportato.

Si agisce dualmente a seconda che la triangola abbia pendenza positiva o negativa, ovvero se la triangola cresce bisogna aprire lo switch alto e chiudere quello basso, mentre quando decresce si deve fare il contrario. Se si era dichiarato un $force\_Hbuf = 1$ allora bisogna controllare quando la triangola raggiunge il valore Mp_force, momento nel quale bisogna chiudere lo switch alto ed eliminare la condizione di forzamento attivata sul vertice. La condizione $force\_Hbuf$ è, infatti, quella di forzamento dovuta al fatto trovarsi sul vertice

basso; a tal proposito si nota che se lo slope è positivo allora si proviene dal vertice negativo.

Si verifica poi che l'Mp di comando sia uguale al valore della triangola e, se la condizione è verificata, si aprono entrambi gli switch; ciò per dare conferma che anche lo switch basso sia aperto.

Infine, se non si è nelle condizioni di forzamento e se lo switch alto è aperto, si controlla quando l'indice dello switch basso diventa pari al valore della portante e quindi lo si chiude.

Analogamente sullo slope negativo.

Implementazione e simulazione del sistema con Modulatore PWM

Per implementare in Vivado il sistema fino ad ora descritto, dopo aver creato un nuovo progetto per l'FPGA di interesse, si aggiungono tra le *sources* i file VHDL relativi ai componenti del sistema.

Una volta creato l'HDL wrapper del main, con l'apposita funzione, è possibile aggiungere il modulo PWM_Modulator al Block Design. Questo è provvisto di una maschera contenente i generic, ovvero delle definizioni a priori comuni a tutti i blocchi interni. Tale maschera, mostrata in Figura 22, permette di cambiare un parametro, come ad esempio il numero di bit del sistema, in modo agevole, senza dover intervenire internamente al codice.

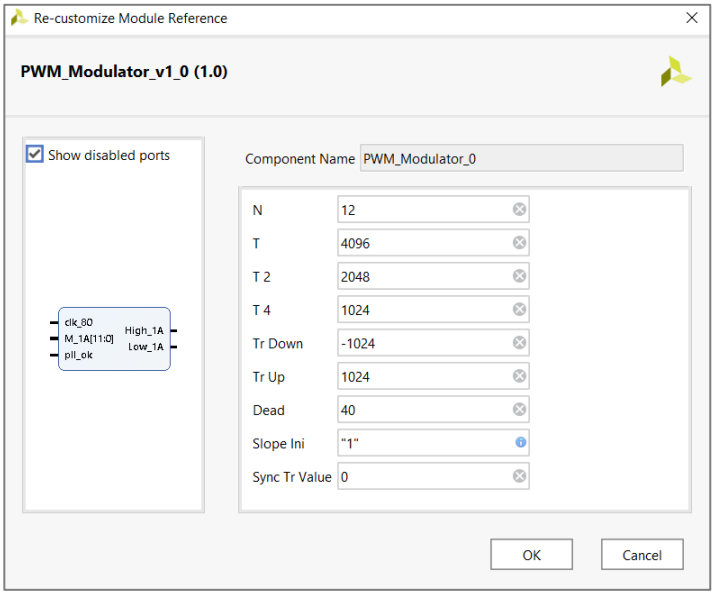


Figura 22 - Generic del Modulatore PWM.

Scegliere di avere un duty cycle con discretizzazione a $N = 12\ bit$ vuol dire che la triangola viene descritta in $2^N = 4096$ colpi di clock, che corrisponde quindi al suo periodo T. Seguono $T/2$ e $T/4$.

Tr_down e Tr_up corrispondono ai valori nei vertici positivo e negativo della triangola.

Per quanto riguarda il dead time questo è messo in relazione al numero di bit, ovvero c'è una relazione tra i colpi di clock e il tempo in µs. Per ottenere un dead time di 1 µs, considerato che questo viene simmetricamente distribuito ai due switch, risulta conveniente calcolare i colpi di clock corrispondenti al parametro $Dead$, pari alla metà del dead time.

$$Dead = \frac{0{,}5\mu s}{T_{clk}} = 0{,}5\mu s \cdot f_{clk} = 40$$

$Slope\ Ini$ indica poi la pendenza iniziale: positiva se 1 e negativa se 0.

$sync\_tr\_value$ corrisponde infine al valore sul quale si vuole sincronizzare il controllo.

Tale blocco va quindi integrato nel sistema con le opportune connessioni, come mostrato in Figura 23.

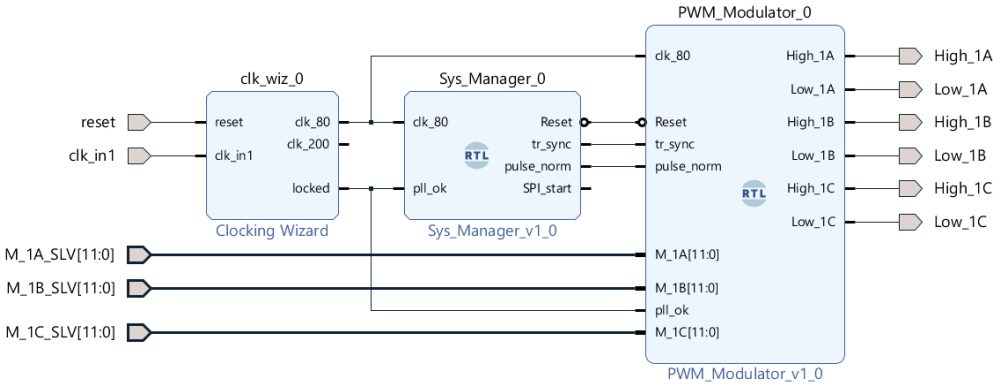


Figura 23 - Sistema con Modulatore PWM.

L'ultimo passo per la validazione del codice è la simulazione, in modo da controllare che i risultati corrispondano con i valori attesi. Si parte con una simulazione di tipo comportamentale, ovvero behavioral. In questa vengono verificati i dead time e la risposta alle variazioni a gradino dell'indice di modulazione quando si entra nelle fasce di dead time.

Prima di procedere alla simulazione si crea il process dell'indice di modulazione per il testbench, di seguito riportato.

```
 stim_proc: process
begin
   -- hold reset state for 100 ns.
   M_1A_SLV <= "000000000000";
   reset <= '1';
   wait for 100 ns;
    reset <= '0';
   wait for clk_125_period*10;
     M_1A_SLV <= "000000000000";
   wait for 200 us;

     M_1A_SLV <= "000000000000";     --duty=50%
   wait for 300 us;
     M_1A_SLV <= "011111111111";     --duty=100%
   wait for 330 us;
     M_1A_SLV <= "100000000001";     --duty=0%
   wait;
end process;
```

Grazie ai testbench creati non è necessario inserire gli stimoli manualmente e si può procedere direttamente alla simulazione.

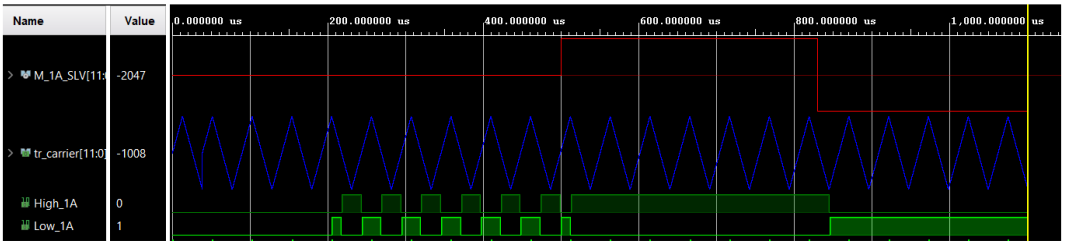


Figura 24 - Behavioral Simulation del sistema con PWM.

In Figura 24 si può notare che i comandi degli switch vengono attuati a partire dal primo vertice (positivo o negativo) successivo alla variazione dell'indice di modulazione.

Il Sistema funziona correttamente anche con variazioni brusche dell'indice di modulazione, come si vede nelle successive tre Figure. In particolare, in Figura 25 si mostra che in una zona di normale funzionamento ($M\_1A\_SLV = 0,\ duty = 0{,}5$) viene rispettata la condizione imposta sul *dead time*. Con le seguenti due figure si verifica inoltre che tale condizione viene rispettata anche all'uscita dalle fasce di dead time, zone che presentano la maggiore criticità.

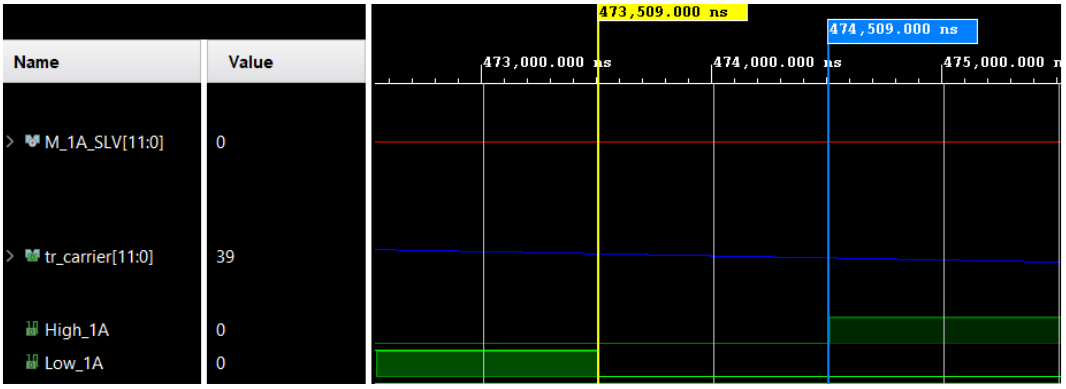


Figura 25 - Verifica dead time per duty=0,5.

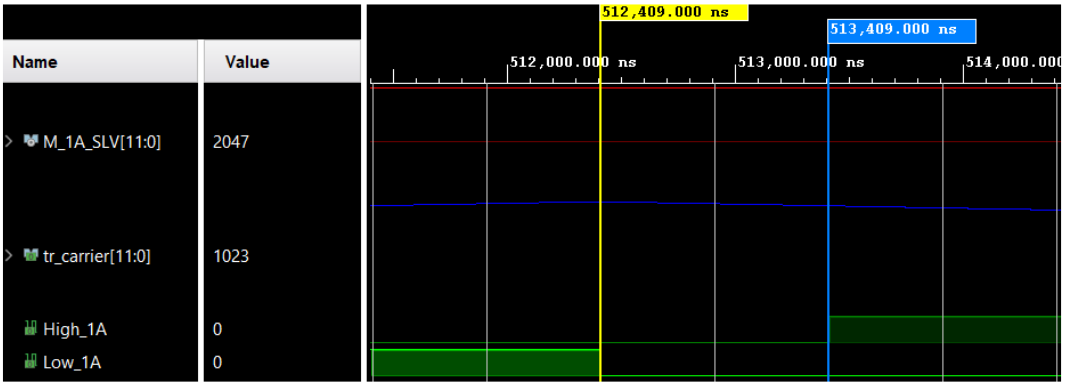


Figura 26 - Verifica dead time per duty=1.

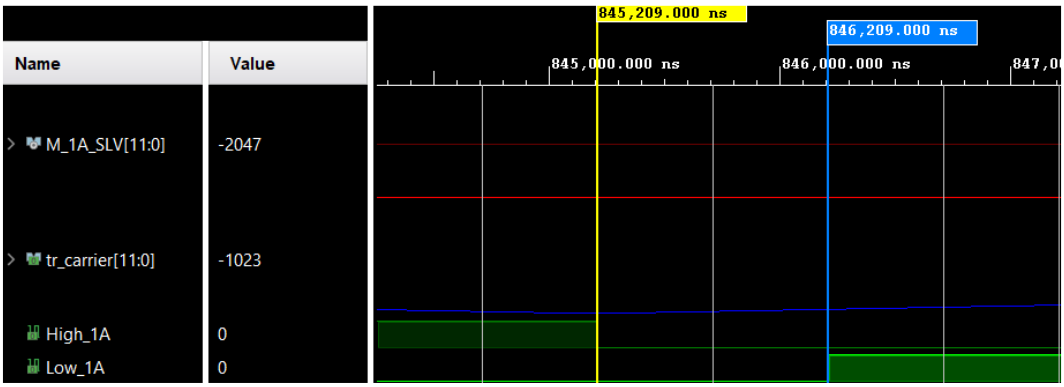


Figura 27 - Verifica dead time per duty=0.

2.3.4 Analog to Digital Converter

Come anticipato, l'ADC è responsabile dell'acquisizione delle grandezze utili al controllo. Il modulo qui implementato riceve le misure dalle schede di misura e le converte così che queste possano essere inviate a dSPACE. È dunque dotato di sette canali di acquisizione, di cui il primo è destinato alla tensione del DC-link, i successivi tre all'acquisizione delle correnti di fase, e gli ultimi tre vengono utilizzati per la terna di tensioni trifase.

Le acquisizioni avvengono quindi tutte in parallelo in modo da ottimizzare i tempi, utilizzando un solo generatore di clock e sette linee seriali.

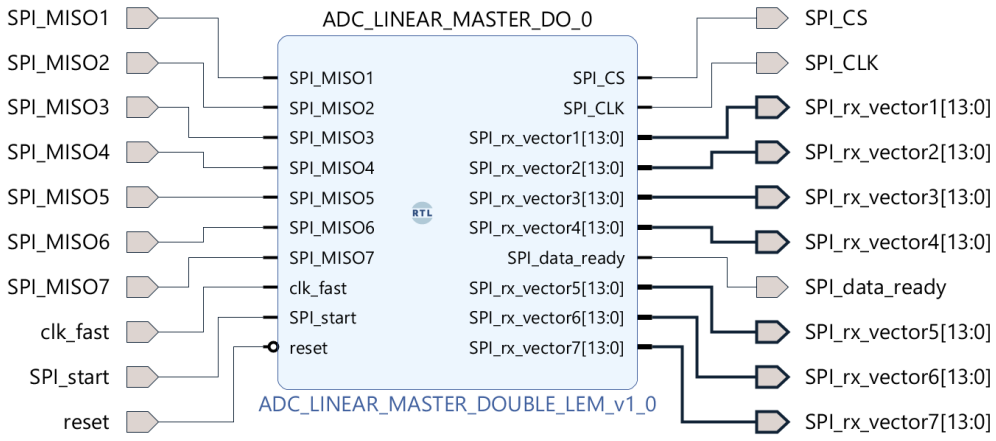


Figura 28 - Analog to Digital Converter.

L'ADC qui implementato necessita di un segnale SPI_start, ovvero un impulso di inizio campionamento che ha lo scopo di avviare l'acquisizione dei dati. Come visto, questo proviene dal System Manager in modo da sincronizzare il campionamento con il vertice alto della portante del modulatore PWM. Quando il segnale SPI_start è pari ad 1 l'acquisizione comincia e il Chip Select (SPI_CS) viene allora portato al livello basso, stato nel quale si abilita la conversione dei segnali. Dal momento in cui SPI_CS è pari a 0, per 16 colpi di SPI_CLK, sul

fronte discendente, si acquisisce un bit della misura per formare la stringa di bit che costituisce l'output.

Il clock in questione è generato internamente a partire dal clock a $200MHz$ che il modulo riceve in ingresso dal Clocking Wizard. A questo viene data una frequenza di $20\ MHz$ impostando il $CLK\_DIVIDER$ dei generic a 5, come mostrato di seguito.

```
generic ( N    : integer := 16;  --number of bit for the communication
  N_ADC        : integer := 14;  --number of bit of ADC converter
  CLK_DIVIDER  : integer := 5;   --the output SPI CLOCK will be
                                   clk_fast/(CLK_DIVIDER*2)
  N_CLK_UP     : integer :=5;    --number of clk_fast cycle to keep
                                   the CS up between two transmission
  CNT          : integer := 8    --counter dimension
 );
```

Nelle righe precedenti si nota che la comunicazione avviene in $16\ bit$, ma i dati trasmessi sono formati solo da $14\ bit$. Esistono infatti tre differenti schemi di temporizzazione:

- SPI_CLK basso durante il tempo di acquisizione, ovvero pari a 0 prima e dopo la comunicazione;
- SPI_CLK alto durante il tempo di acquisizione, ovvero pari ad 1 prima e dopo la comunicazione;
- SPI_CLK continuo, ovvero con il clock attivo che alterna gli stati alto e basso durante tutto il processo.

Di questi, quello qui implementato è il primo e viene mostrato in Figura 29. La transizione dello SPI_CS al livello basso rimuove il forzamento dell'uscita dallo stato logico alto, per cui da quel momento sui fronti discendenti del clock vengono acquisiti i dati dal MSB al LSB. Il primo e l'ultimo fronte di discesa non corrispondono però all'acquisizione di dati e sono rispettivamente chiamati *leading* e *trailing zeroes*. Mentre col primo schema si ha un solo *leading zero*, si nota che con gli altri due ne vengono prodotti due. Il flusso di dati risulta quindi costituito da uno o due zeri iniziali, seguiti dai $14\ bit$ di dati di conversione. In tutti i casi questi zeri permettono di inquadrare il risultato in 14 bit per la verifica dei tempi e dei dati. Il processo si arresta dopo il sedicesimo fronte discendente del clock, quando il segnale SPI_CS torna ad 1 e l'uscita viene nuovamente forzata allo stato logico alto.

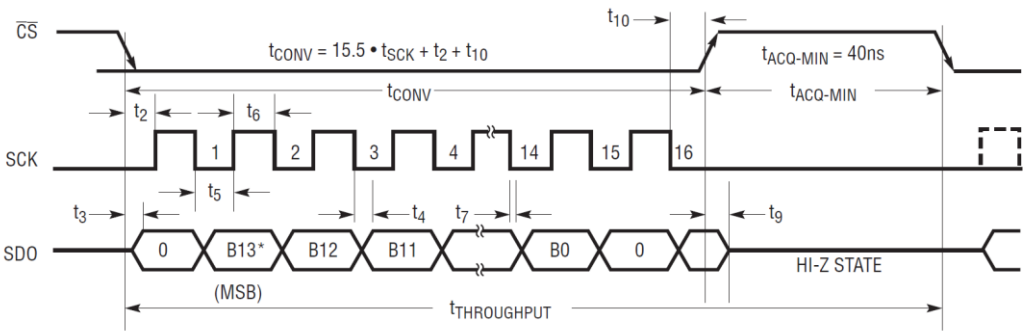


Figura 29 - Diagramma timing dell'interfaccia seriale [15].

I tempi t_2 e t_{10}, mostrati nella Figura precedente sono rispettivamente il ritardo tra il fronte di discesa del Chip Select e il fronte di salita SPI_CLK e il ritardo tra il sedicesimo fronte di discesa del clock e quello di salito dello SPI_CS. Questi devono essere osservati affinché il processo di conversione e la lettura dei dati avvengano correttamente.

Implementazione e simulazione del sistema con ADC

Si verifica il funzionamento in simulazione. In particolare si intende verificare:

- La sincronizzazione del segnale SPI_start rispetto al vertice della triangola portante del modulatore PWM;
- La sequenza di SPI_CS;
- La sequenza di SPI_CLK;
- Il segnale di SPI_data_ready.

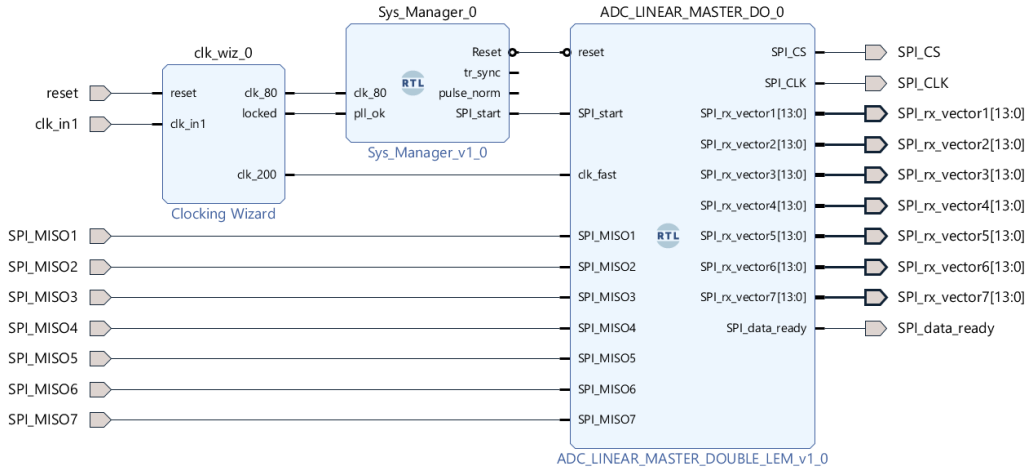


Figura 30 - Sistema con ADC.

È cioè necessario verificare che il sistema invii l'impulso di inizio campionamento nel momento desiderato e che, in seguito a questo il Chip Select si porti allo stato basso. Nell'intervallo di tempo in cui SPI_CS è pari a zero il

clock dovrà essere abilitato e segnare sedici colpi. Si sottolinea ora che il dispositivo prevede un ciclo di latenza prima di rendere i dati disponibili, per cui il segnale SPI_data_ready che indica la fine del processo dovrà presentare un impulso alla fine del secondo ciclo.

Si crea quindi un testbench di simulazione per emulare i segnali SPI_MISO in ingresso. Per semplicità si generano degli ingressi ad onda quadra nei quali si alternano stati alti e bassi.

```
SPI_MISO : process(SPI_CS, SPI_CLK, Reset)
    begin
    if(Reset = '1') then

        SPI_MISO1 <= '0'; --inizializza
        SPI_MISO2 <= '1';
        SPI_MISO3 <= '1';
        SPI_MISO4 <= '0';
        SPI_MISO5 <= '1';
        SPI_MISO6 <= '1';
        SPI_MISO7 <= '1';

    elsif (SPI_CS = '0') then
        if(falling_edge(SPI_CLK)) then
            SPI_MISO1 <= not SPI_MISO1;
            SPI_MISO2 <= not SPI_MISO2;
            SPI_MISO3 <= not SPI_MISO3;
            SPI_MISO4 <= not SPI_MISO4;
            SPI_MISO5 <= not SPI_MISO5;
            SPI_MISO6 <= not SPI_MISO6;
            SPI_MISO7 <= not SPI_MISO7;
          end if;
    end if;
end process;
```

Nella Figura seguente è possibile verificare che il sistema operi come richiesto. Ottenendo infine sette stringhe di dati che contengono l'acquisizione di tutte le grandezze di interesse.

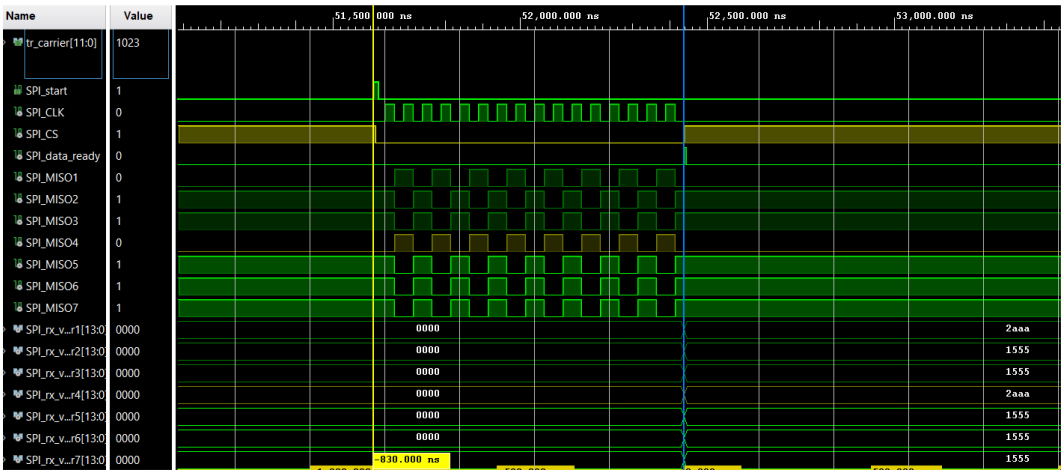


Figura 31 - Behavioral simulation del sistema con ADC.

Si nota che un ciclo di campionamento avviene in poco più di 800 ns. Questo costituisce un ottimo risultato poiché i dati in uscita dal modulo devono essere

inviati a dSPACE, che li processa e rimanda indietro all'FPGA i valori aggiornati dei duty cycle. Tali valori devono dunque essere disponibili entro il vertice successivo della triangola, momento nel quale il modulatore PWM comincia i calcoli necessari a produrre i nuovi segnali di gamba per gli switch dell'inverter. Il tempo di campionamento e della trasmissione dati deve quindi essere il più ridotto possibile per garantire che dSPACE abbia sufficiente margine per eseguire il controllo. Si sottolinea comunque che in questo caso specifico il tempo di acquisizione non costituisce una criticità poiché il controllo viene implementato ad una frequenza di 10 kHz (100 μs), quindi il campionamento e il trasferimento dei dati richiede addirittura meno dell'1% del tempo totale a disposizione.

ADC SPI isolato

Sulle schede di acquisizione CAB e VAB sono presenti degli isolatori che isolano galvanicamente il circuito di misura da quello in cui sono presenti i segnali che vengono scambiati con la scheda FPGA, come mostrato in Figura 32.

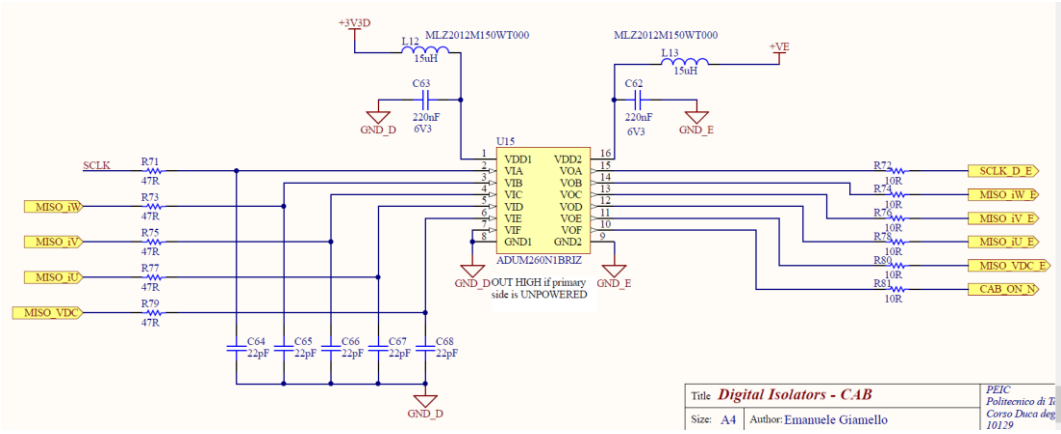


Figura 32 - Isolatore ADuM260N1BRIz.

Questi sono basati sul principio trasformatorico ed introducono un ritardo di propagazione, che è un'indicazione della precisione con cui viene preservata la temporizzazione del segnale di ingresso. In Figura 33 si mostra lo schema dell'isolatore utilizzato.

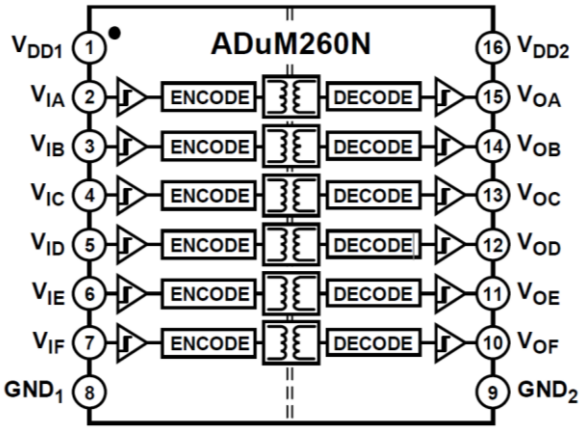


Figura 33 - ADuM260N1 [4].

I segnali in uscita dal componente sono quindi in ritardo rispetto a quelli in ingresso di un tempo che, secondo il datasheet presente in bibliografia [15], è pari a $15\,ns$. In corrispondenza del segnale SPI_CLK inviato dall'FPGA si genera dunque un secondo clock, che prende il nome di SPI_CLK_D, in ritardo rispetto al primo e sul quale avviene il campionamento delle grandezze. I segnali di acquisizione che vengono inviati all'FPGA, devono essere quindi sincronizzati al secondo clock, ovvero quello presente sul lato del circuito di misura, e devono poi attraversare l'isolatore risentendo a loro volta del ritardo e risultando in definitiva $30\,ns$ in ritardo rispetto a quelli che si sarebbero ottenuti in assenza del componente ritardante.

Per questo motivo se si sincronizzasse la lettura dei dati sul clock di andata si commetterebbe un errore proporzionale allo sfasamento dei due clock. In Figura 34 si mostrano i suddetti clock quando si imposta da FPGA una frequenza di $25MHz$, che corrisponde ad un periodo di $40\,ns$. Poiché il ritardo complessivo risulta prossimo a tale periodo si osservano i clock circa in controfase. In Figura 35 si presenta il risultato corrispondente alla frequenza di $10\,MHz$ (periodo pari a $100\,ns$), per cui si osserva uno sfasamento di circa 1/3 del periodo.

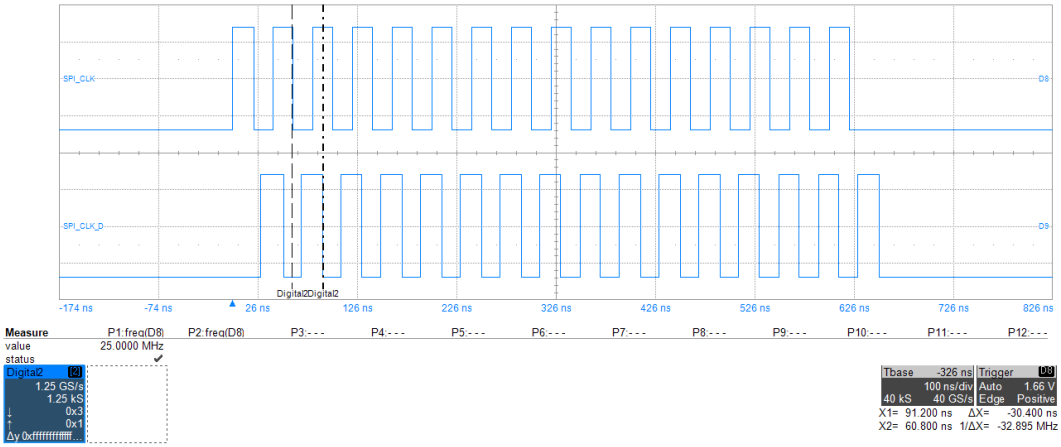


Figura 34 - SPI_CLK in alto e SPI_CLK_D in basso, $f = 25\,MHz$.

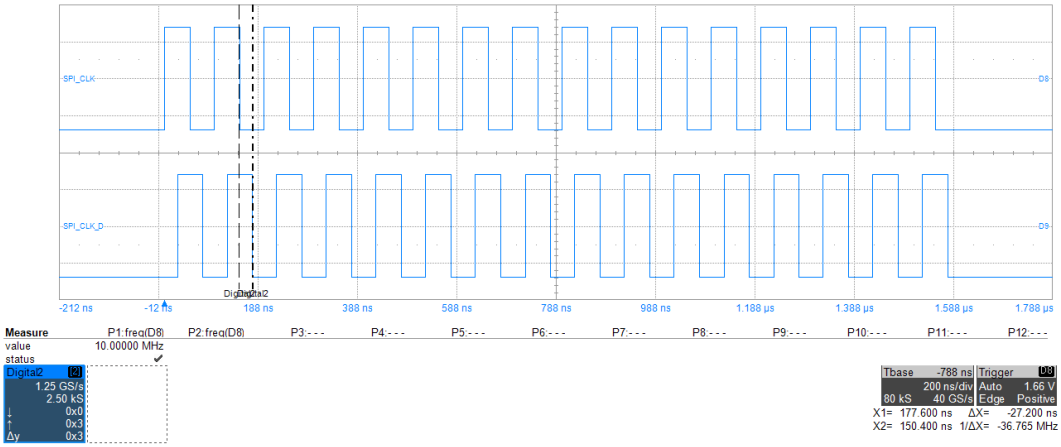


Figura 35 - SPI_CLK in alto e SPI_CLK_D in basso, $f = 10\,MHz$.

Sincronizzando le acquisizioni al clock di ritorno è possibile lavorare a frequenze più alte, compatibilmente con i limiti degli ADC a bordo delle schede, senza incorrere nell'errore appena descritto. Di seguito si riportano alcuni risultati ottenuti impostato da FPGA la frequenza dello SPI_CLK pari a $50\ MHz$.

In Figura 36 è rappresentato dall'alto verso il basso il chip select, il segnale MISO, il clock generato dal modulo ADC e, infine, il clock di ritorno in ritardo rispetto al primo.

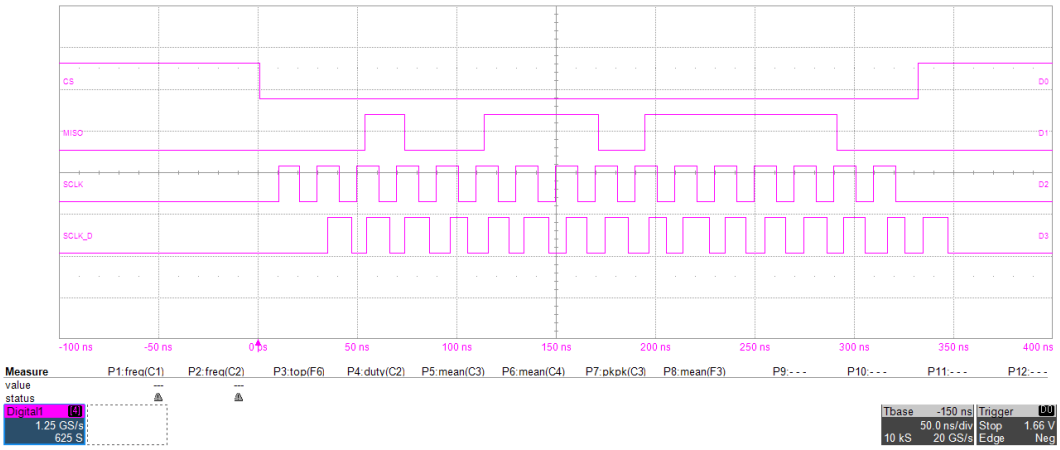


Figura 36 - Segnali di acquisizione SPI.

In Figura 37 si confrontano i due clock, di andata e ritorno, con periodo di $20\ ns$ e ritardo di $30\ ns$.

In Figura 38 si mostra infine che la sincronizzazione dell'acquisizione avviene correttamente con il clock di ritorno SPI_CLK_D.

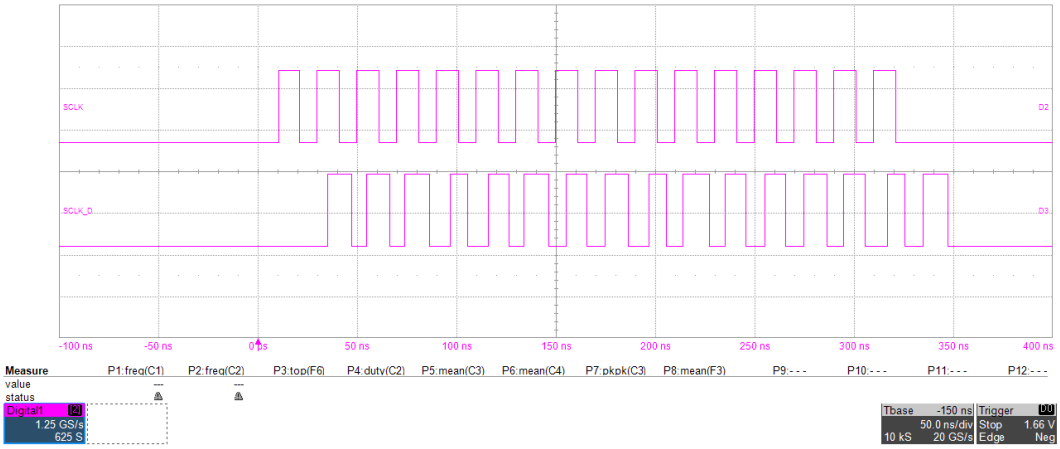


Figura 37 - Sfasamento SPI_CLK e SPI_CLK_D.

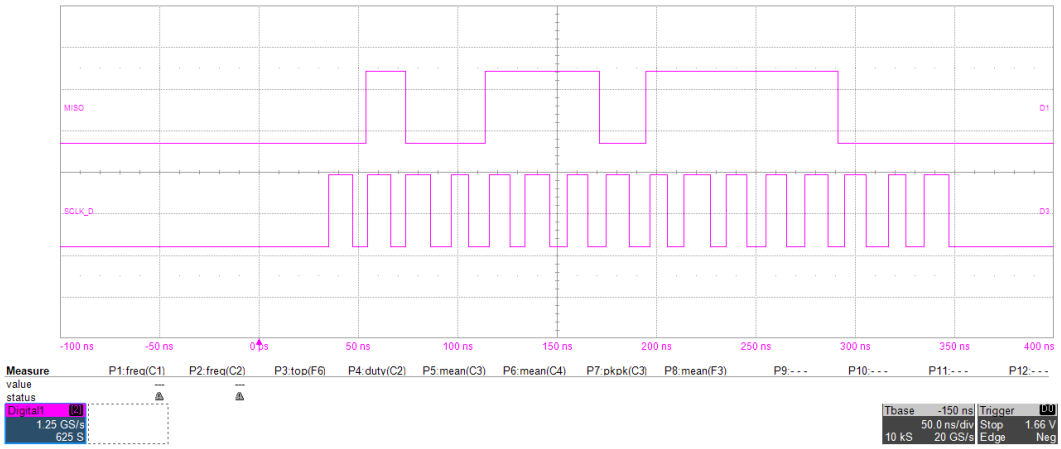


Figura 38 - Sincronizzazione MISO con SPI_CLK_D.

2.3.5 Protezioni hardware analogiche

Le protezioni hardware analogiche riguardano la messa in sicurezza in risposta ad eventuali sovracorrenti o sovratensioni nelle grandezze dell'inverter. È fondamentale che il sistema sia capace di rilevare la presenza di un guasto così da riconoscere la necessità di proteggersi da ogni malfunzionamento. Il modulo di protezione qui implementato ha proprio la funzione di rilevare la presenza di un guasto e generare gli opportuni segnali di errore.

La logica utilizzata è quella di introdurre delle soglie che limitano i valori che le misure acquisite possono assumere, superate le quali si ritiene che sull'inverter vi sia la presenza di un guasto. Si ricorda che tali misure sono: la tensione del DC-link, la terna di correnti e la terna di tensioni trifase sulle tre fasi. Gli ingressi sono quindi costituiti dai segnali campionati a $14\,bit$ provenienti dal modulo di gestione ADC, come mostrato in Figura 39.

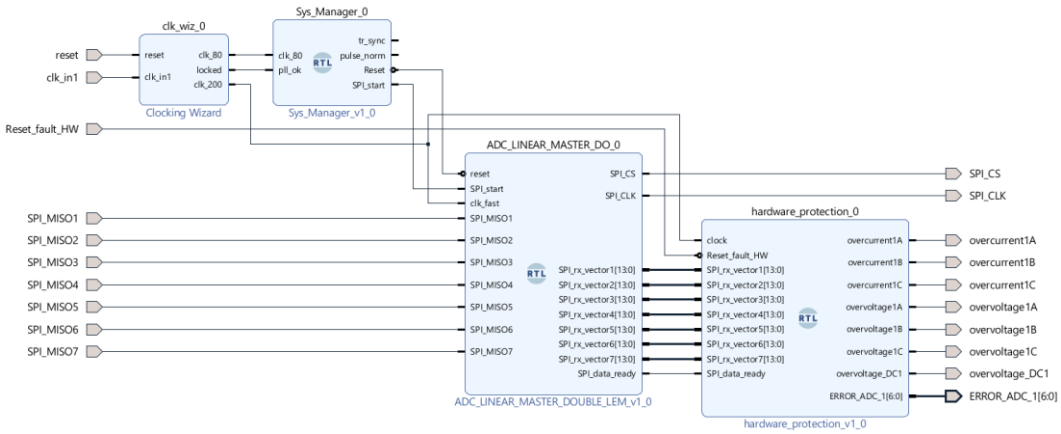


Figura 39 - Sistema con protezione hardware analogiche.

Questi sono tutti segnali unsigned compresi tra due livelli, cioè 0 e $2^{14} - 1$, ma mentre la tensione del DC-link è una misura unipolare, le altre sei grandezze,

in quanto sinusoidi, sono bipolari. Si sottolinea che le soglie e il valore di 0 dipendono dal circuito di condizionamento. In particolare, il valore dello 0 dipende dall'offset del circuito, il quale viene identificato in fase di commissioning.

Per quanto riguarda la protezione da sovratensione sul DC-link è infatti sufficiente introdurre un'unica soglia superiore, superata la quale si genera l'opportuno segnale di errore. Con riferimento alla Figura successiva si spiega la logica di implementazione.

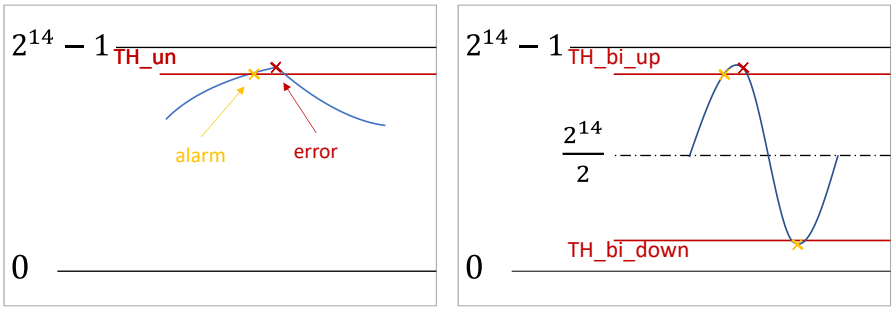


Figura 40 - Logica protezioni hardware analogiche: misure unipolari a sinistra e bipolari a destra.

In corrispondenza della prima misura fuori soglia, indicata con la X gialla, si genera un segnale di allarme. Nel caso in cui segua una seconda misura fuori soglia allora si invia l'opportuno segnale di errore, ma se la grandezza misurata rientra entro i limiti impostati allora si resetta l'allarme prima generato e non viene inviato alcun segnale di guasto. Questo serve ad evitare, ad esempio, problemi di falso campionamento, disturbi e rumori.

Per quanto riguarda le misure bipolari si osservi che lo 0 si trova a metà scala, cioè in corrispondenza di $2^{14}/2$, per cui si rende necessario definire due soglie per proteggersi da picchi positivi e negativi, come mostrato in Figura 40.

Implementazione e simulazione del sistema con protezioni hardware analogiche

Si mostrano di seguito i risultati della simulazione comportamentale del sistema nel quale è stato inserito il modulo di protezione hardware. Si noti come prima dell'acquisizione, quando tutte le misure sono nulle, sui canali di corrente e tensione AC vi è un segnale di errore, in quanto la misura è nulla e quindi inferiore alla soglia down impostata. Sul canale della tensione DC invece, dove è presente solo un limite superiore, l'errore è sempre nullo.

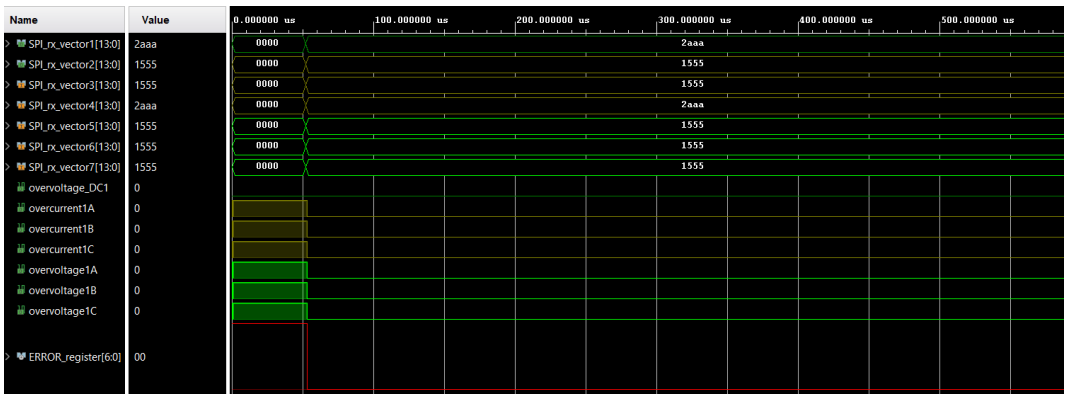


Figura 41 - Behavioral simulation del sistema con protezioni HW analogiche.

Il modulo produce in uscita l'ERROR_register, ovvero una stringa di dati in cui ad ogni bit è associato un tipo di errore. Questo risulterà utile nel seguito quando si organizzeranno i dati in registri.

2.3.6 Protezioni IGBT

L'FPGA riceve dalla scheda IIB un segnale di *fault*, che indica la presenza di un malfunzionamento sulla scheda driver, ad esempio un problema all'alimentazione, oppure un guasto come un cortocircuito. Questo segnale ha la funzione di avviare l'apertura degli switch così da proteggerli dalla circolazione di correnti di cortocircuito. Il modulo di protezione IGBT qui implementato serve proprio a comandare l'apertura degli interruttori in risposta al segnale di $Fault\_IGBT$ in ingresso.

Una seconda condizione che permette di inviare segnali di gamba non nulli al convertitore è l'abilitazione proveniente da dSPACE, cioè il segnale PWM_EN. Se per qualche ragione, infatti, i driver si accendessero l'assenza di questo segnale impedirebbe al sistema di produrre i comandi degli switch senza che il controllo in dSPACE abbia mandato l'abilitazione.

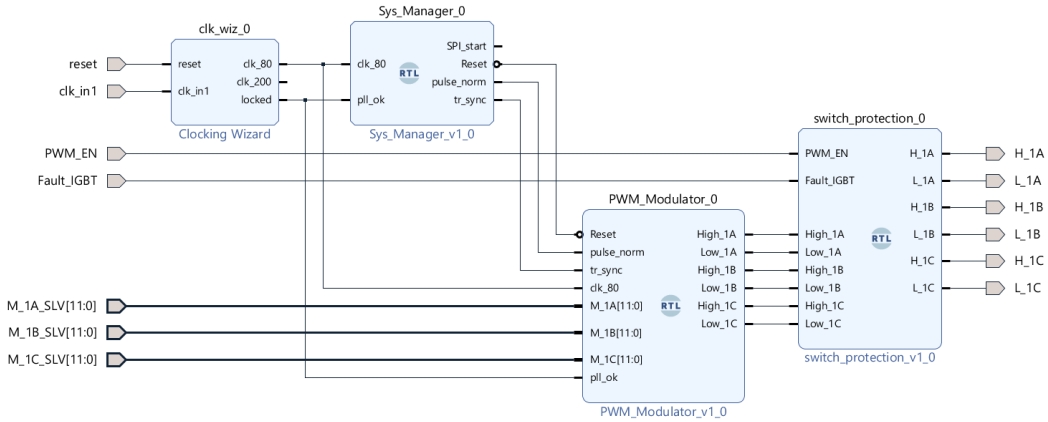


Figura 42 - Sistema con protezioni IGBT.

Una volta generati i segnali degli switch col PWM_Modulator, questi vengono mandati in ingresso al blocco di protezione che assicura che gli interruttori siano sempre aperti per qualsiasi valore di comando di gate in caso di guasto o mancato segnale di Enable. Il blocco riceve quindi in ingresso i sei segnali degli switch e PWM_EN e $IGBT\_Fault$, e produce in uscita i segnali di gate per la scheda interfaccia inverter.

Il modulo viene realizzato con porte logiche così da essere eseguito in continuazione, anche senza sincronizzazione. Le porte logiche vengono infatti eseguite istantaneamente, mentre la scrittura di un codice sincronizzato ad un clock in ingresso, comporterebbe un ritardo di almeno un colpo di clock; con le porte logiche invece la propagazione dipende solo dal tempo di propagazione.

Tabella 1 - Tabella di verità del blocco Protezioni IGBT.

$Gate\_Q$	PWM_EN	$Fault$	$Gate\_Q\_IIB$
X	0	0	0
X	1	0	$Gate\_Q$
X	0	1	0
X	1	1	0

Si implementa quindi il sistema in Figura 43 per verificare la tavola di verità della Tabella 1, dove l'uscita è data da:

$$Gate\_Q\_IIB = (PWM\_EN \cdot \overline{Fault_{IBGT}}) \cdot Gate\_Q$$

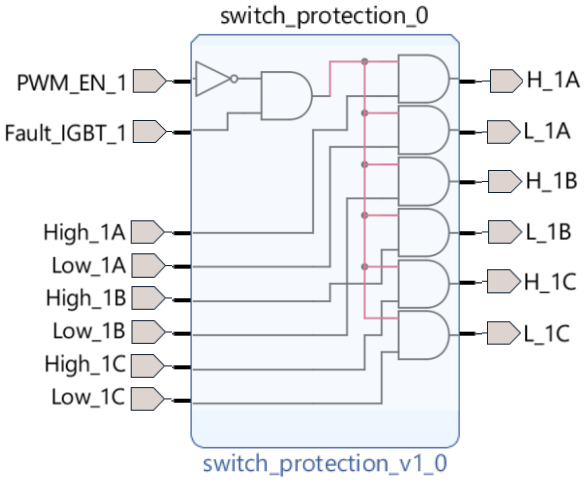


Figura 43 - Logica del blocco Protezioni IGBT.

Implementazione e simulazione del sistema con protezioni IGBT

Per verificare il corretto funzionamento del modulo, si scrive l'opportuno codice testbench, nel quale si portano rispettivamente a 0 e ad 1 i segnali PWM_EN e $Fault\_IGBT$ in maniera periodica. Di seguito si riportano i risultati della simulazione per entrambi i comandi di tutte e tre le gambe dell'inverter. In Figura 44 si verifica facilmente che il comando in uscita è uguale a quello in ingresso, a meno che il segnale di abilitazione non sia nullo o il segnale di guasto non valga 1.

In Figura si osserva che quando il PWM_EN vale 0, tutti i comandi di gamba si annullano fino a ché il modulo non riceve nuovamente l'abilitazione. Analogamente i segnali per la scheda IIB si annullano istantaneamente quando il $Fault\_IGBT$ è diverso da 0.

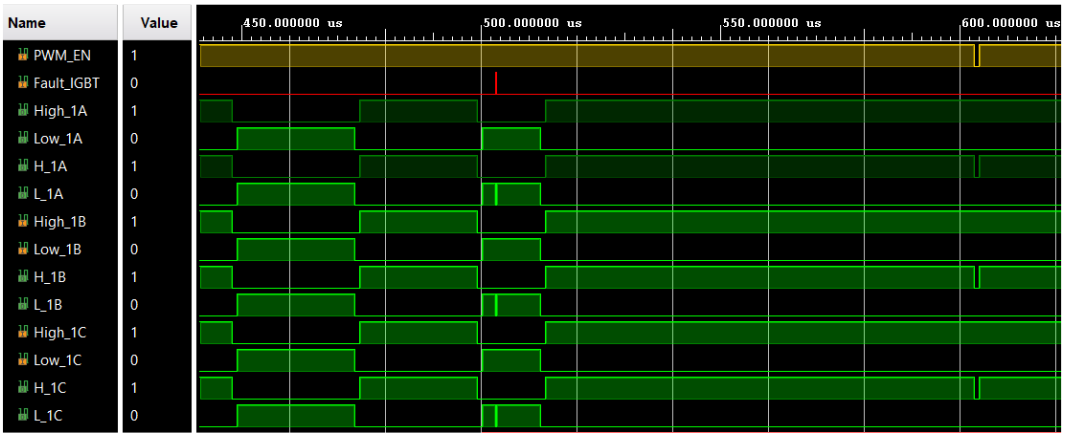


Figura 44 - Behavioral simulation del sistema con Protezioni IGBT.

2.3.7 Digital to Analog Converter

Si utilizza un Digital to Analog Converter, di cui si riporta il riferimento in bibliografia [8], con funzione di debug. Questo modulo fornisce infatti la possibilità di verificare che le acquisizioni dell'ADC avvengono correttamente se dalla riconversione si ottiene in uscita lo stesso dato e consente, inoltre, di visualizzare le grandezze acquisite sull'oscilloscopio così da verificare che tutto funzioni correttamente.

Il modulo implementato consente la gestione di un DAC a quattro canali che riceve quattro registri dati e fornisce quattro uscite analogiche contemporanee.

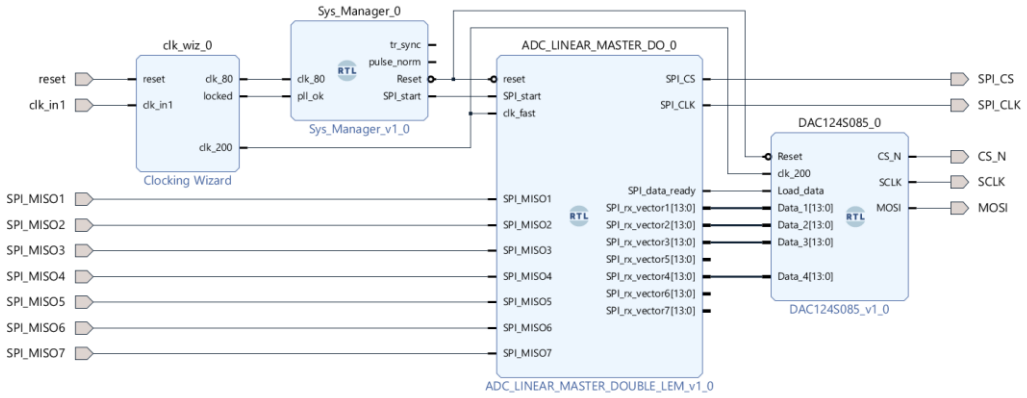


Figura 45 - Sistema con DAC.

La logica di funzionamento è analoga a quella del modulo di gestione dell'ADC. Quando il segnale di *load_data*, corrispondente al segnale di *SPI_data_ready* del modulo ADC, viene generato allora vengono aggiornati i dati sui quattro canali del DAC e si avvia la trasmissione. Il Chip Select si porta dunque allo stato basso per 16 colpi di clock, che sono infatti i bit di trasmissione. Il clock in questione è nuovamente generato internamente al blocco, grazie alla definizione di un clock divider fra i generic, a partire dal clock veloce a 200 MHz in uscita dal Clocking Wizard. Dal momento in cui il Chip Select va a 0 in corrispondenza dei fronti discendenti di *SCLK* si acquisisce il segnale sulla linea MOSI.

Come detto, la trasmissione avviene su 16 bit, ma solo 12 sono riservati alle misure, i primi due infatti indicano l'indirizzo del canale ed i successivi due la modalità di funzionamento. Si nota che, poiché solo 12 bit sono riservati alle misure e i dati in ingresso sono invece a 14 bit, è necessario operare uno shift eliminando i 2 Least Significant Bit.

Dopo il sedicesimo colpo di clock il Chip Select si riporta allo stato alto e l'acquisizione del primo dato termina. Il processo viene quindi ripetuto in maniera analoga per la conversione dei successivi tre dati.

Implementazione e simulazione del sistema con DAC

Di seguito si presentano i risultati della simulazione comportamentale del sistema comprendente il modulo Digital to Analog Converter.

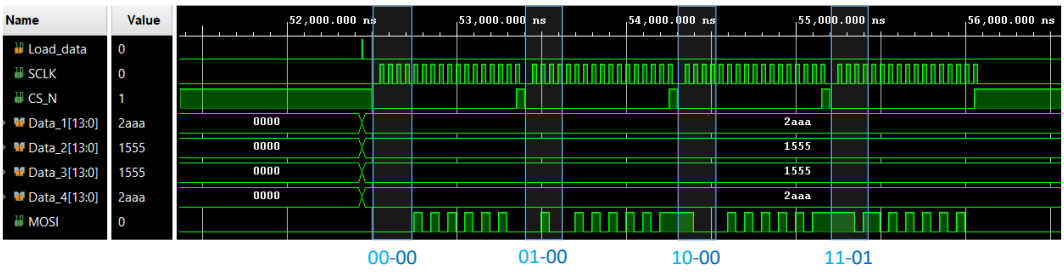


Figura 46 - Behavioral simulation del sistema con DAC.

Quando i dati dell'ADC sono pronti, cioè arriva l'impulso di $Load\_data$, il DAC inizia la sua conversione portando a 0 il Chip Select per quattro cicli successivi da 16 colpi di clock, ognuno dei quali corrisponde all'acquisizione di un dato. In Figura 46 si evidenziano i primi quattro bit di ciascun dato presente sulla linea MOSI, ovvero gli indirizzi da 0 a 3 corrispondenti ai quattro canali e le modalità 00, ovvero di sola lettura e 01, che indica di scrivere e aggiornare tutte le uscite. Le modalità 10 e 11 corrispondono rispettivamente alla scrittura su tutti e quattro i canali e l'aggiornamento di tutti gli output oppure allo spegnimento di tutti e quattro gli output.

Il processo di trasmissione termina quando il Chip Select torna allo stato logico alto per la quarta volta.

2.3.8 Finite State Machine

Per la gestione degli stati dell'inverter si implementa una Finite State Machine, che riceve i segnali di stato da dSPACE. La macchina parte dallo stato ERROR per passare al successivo stato di RESET quando il corrispondente segnale viene ricevuto. In questo stato vengono resettati gli errori, ma il sistema non è ancora pronto ad operare. Quando dSPACE invia il segnale di inizializzazione la macchina passa allo stato READY, in cui avvengono tutte le operazioni di inizializzazione, quale ad esempio la precarica del DC-link. Una volta che il convertitore è pronto, il segnale di abilitazione viene inviato e si passa allo stato GO, corrispondete al normale funzionamento. Un qualunque segnale di malfunzionamento provoca infine il ritorno della macchina allo stato iniziale di errore, dal quale l'uscita è condizionata dalla risoluzione del suddetto problema e avviene ripercorrendo gli stati precedentemente elencati. La logica implementata è mostrata nella Figura 48.

Il modulo implementato in Vivado è mostrato in Figura 47. Questo riceve in ingresso il clock veloce a 200 MHz proveniente dal Clocking Wizard, un registro di configurazione contenente gli stati della macchina, proveniente da dSPACE, e i segnali di guasto provenienti dall'inverter e dal modulo di protezione hardware. L'uscita è un comando di errore globale che ha la funzione di mettere il sistema in uno stato sicuro.

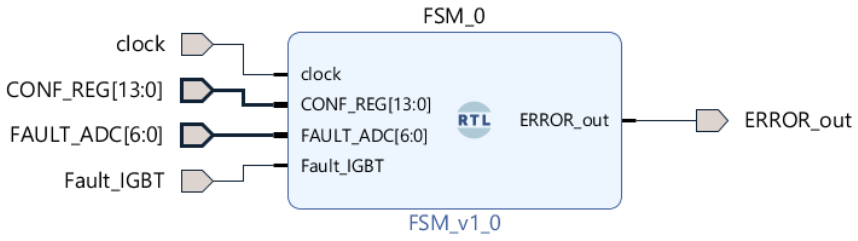


Figura 47 - Finite State Machine.

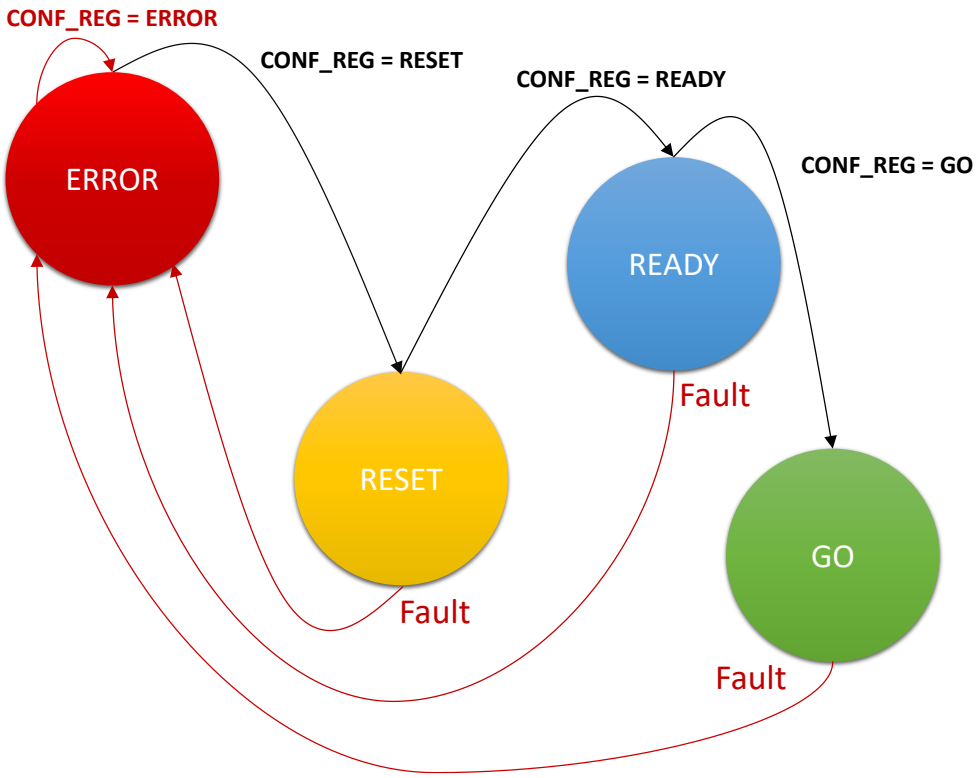


Figura 48 - Diagramma Macchina a Stati.

Per la scrittura del codice del modulo in VHDL si crea il tipo di dato “state” con la funzione *type* e a questo si assegna la lista degli stati: ERROR, RESET, READY, GO. Si genera poi un segnale di tipo stato come mostrato dalle seguenti righe di codice.

```
architecture Behavioral of FSM is
type state is (ERROR, RESET, READY, GO);
signal state_curr : state;
signal state_next : state;
signal ERROR_in : std_logic;
signal RESET_in : std_logic;
signal READY_in : std_logic;
signal GO_in : std_logic;
signal FAULT : std_logic;
```

Per la gestione degli avanzamenti di stato e delle azioni corrispondenti ad ogni stato si genera un process. Si avanza quindi di stato in accordo al registro configurazioni e si torna nello stato ERROR nel caso in cui il segnale *FAULT*, nato dalla combinazione dei segnali di guasto in ingresso, assume il valore di 1 logico.

```
NEXT_STATE_DECODE : process(state_curr, FAULT_ADC, Fault_IGBT)
  begin
       if(FAULT_ADC /= "0000000" or Fault_IGBT = '1') then
         FAULT <= '1';
       end if;
```

```
    case (state_curr) is
     when ERROR => if (ERROR_in = '1') then
                     state_next <= ERROR;
                   else if (RESET_in= '1') then
                     state_next <= RESET;
                   end if;
                   end if;
                   ERROR_out <= '1';
     when RESET => if (RESET_in = '1') then
                     state_next <= RESET;
                   else if (READY_in = '1') then
                     state_next <= READY;
                   end if;
                   end if;
                     if (FAULT = '1') then
                     state_next <= ERROR;
                     end if;
                   ERROR_out <= '0';
     when READY => if (READY_in = '1') then
                     state_next <= READY;
                   else if (GO_in = '1') then
                     state_next <= GO;
                   end if;
                   end if;
                     if (FAULT = '1') then
                     state_next <= ERROR;
                     end if;
                   ERROR_out <= '0';
     when GO    => if (GO_in = '1') then
                     state_next <= GO;
                   end if;
                     if (FAULT = '1') then
                     state_next <= ERROR;
                     end if;
                   ERROR_out <= '0';
    end case;
  end process;
```

Implementazione e simulazione del sistema con Finite State Machine

Di seguito si riportano i risultati della simulazione comportamentale del sistema in cui è stato annesso il modulo FSM.

Dal momento in cui il segnale $FAULT$, realizzato come intersezione dei segnali $FAULT\_ADC$ e $Fault\_IGBT$, assume il valore 0, la macchina a stati può lasciare lo stato di ERROR. Questa avanza fino allo stato GO di operatività in accordo al registro configurazione che riceve da dSPACE. In Figura 49 si può infatti verificare che quando il modulo riceve il segnale in codice esadecimale 0002 allora la macchina passa al corrispondente stato di RESET, analogamente quando il registro di configurazione riporta il valore 0004 la macchina si porta in READY e, infine, allo stato GO corrispondente all'ingresso 0008. Si nota inoltre che il segnale di errore globale in uscita si annulla quando in ingresso non vengono registrati errori provenienti dall'inverter o dalle protezioni hardware.

Figura 49 - Behavioral simulation del sistema con FSM.

2.3.9 Comunicazione con dSPACE

La scheda FPGA e dSPACE scambiano dati con una comunicazione parallela in cui sono disponibili fino a $24\ bit$ (da 0 a 23) sia in ingresso che in uscita. In Figura 50 si mostrano gli schemi dei connettori dell'FPGA di andata (a sinistra) e ritorno (a destra) da dSPACE verso l'FPGA.

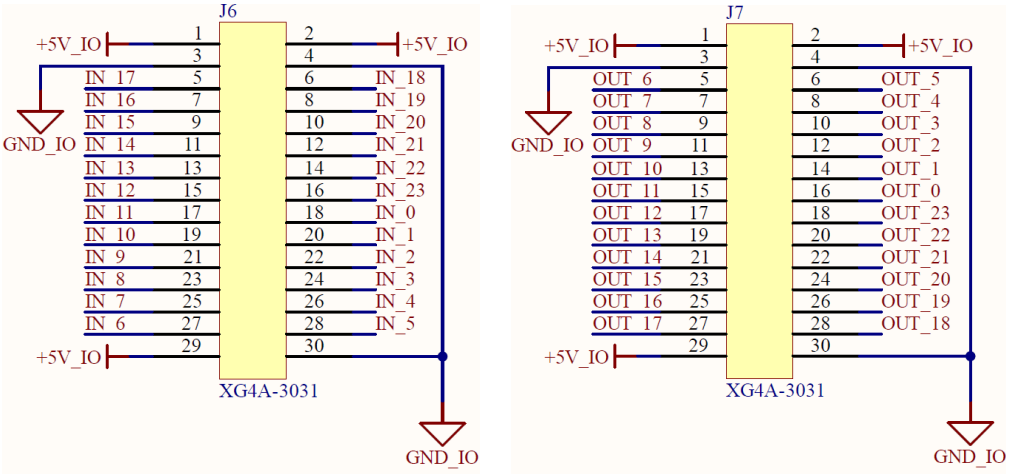


Figura 50 - Connettori della scheda FPGA per il collegamento con dSPACE.

Nel corso del presente capitolo sono stati nominati alcuni dei dati che vengono ricevuti o mandati a dSPACE, sul quale viene implementato il codice di alto livello per la generazione dei duty cycle degli inverter Semikron. Suddividendo i dati di interesse come mostrato in Figura 51 si contano dieci segnali in uscita:

- Otto acquisizioni analogiche, ovvero tre correnti per ciascun inverter e le due tensioni di DC-link;
- Un registro errori hardware, generato dal modulo di protezione HW;
- Un segnale IRQ, che rappresenta l'interrupt del codice C di dSPACE.

Altri dieci segnali viaggiano invece nella direzione opposta, da dSPACE verso l'FPGA, e sono:

- Sei duty cycle, ovvero uno per ogni gamba di ciascun inverter trifase;
- Un registro configurazioni, contenente i segnali di abilitazione della modulazione dei due blocchi PWM_Modulator, il segnale di

$Fault\_IGBT$ in ingresso al modulo di protezione switch, il segnale $Reset\_HW$ per il modulo di protezione HW ed un segnale di abilitazione per le schede VAB di cui si parlerà al paragrafo 4.2;

- Tre segnali di comunicazione, ovvero EOP, RD_WR e $REQUEST$.

Con riferimento alla Figura 51 si osserva che i dati in ingresso all'FPGA sono costituiti da $12\,bit$, mentre quelli in uscita da $14\,bit$. Ciò è dovuto alle impostazioni con le quali è stato pensato il codice, infatti il modulatore lavora in funzione di un numero Nbit impostabile dall'utente, che è posto pari a 12. Il registro configurazioni raccoglie in effetti meno di dodici segnali, e quindi potrebbe essere ridotto, ma si sceglie di lasciare i bit extra a disposizione per eventuali utilizzi futuri. Le acquisizioni analogiche avvengono per mezzo di ADC montati su scheda [15] che funzionano a $14\,bit$. Anche il modulo di gestione viene allora fatto lavorare a $14\,bit$ e, dal momento che per la comunicazione sono disponibili molti pin sugli appositi connettori, si decide di mantenere la migliore risoluzione possibile. Il registro errori potrebbe anche esso essere ridotto, ma come per il registro configurazioni si lasciano i bit inutilizzati a disposizione. Si sottolinea infine che i segnali IRQ, EOP, RD_WR e $REQUEST$ viaggiano in parallelo su bit dedicati, mentre gli altri dati vengono mandati e ricevuti in sequenza, sfruttando quindi un totale di $15\,bit$ totali per ciascuna direzione della comunicazione.

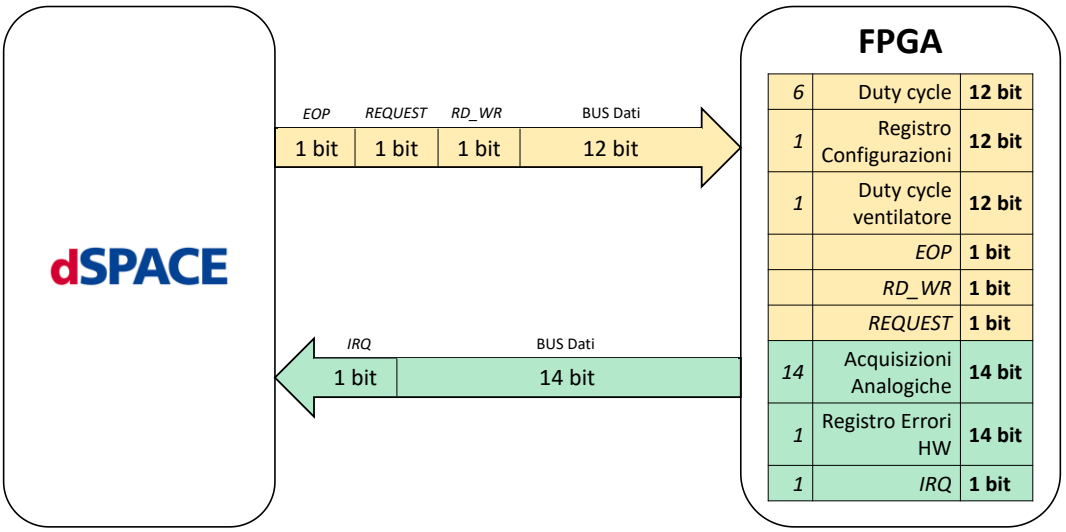


Figura 51 - Schema di comunicazione FPGA-dSPACE.

Si implementa dunque un modulo dedicato alla comunicazione, mostrato in Figura 52, che riceve in ingresso un vettore con tutti i dati provenienti da dSPACE, denominato DOFI (dSPACE Output FPGA Input), e genera in uscita un vettore DIFO (dSPACE Input FPGA Output) con i dati utili al controllo implementato su dSPACE.

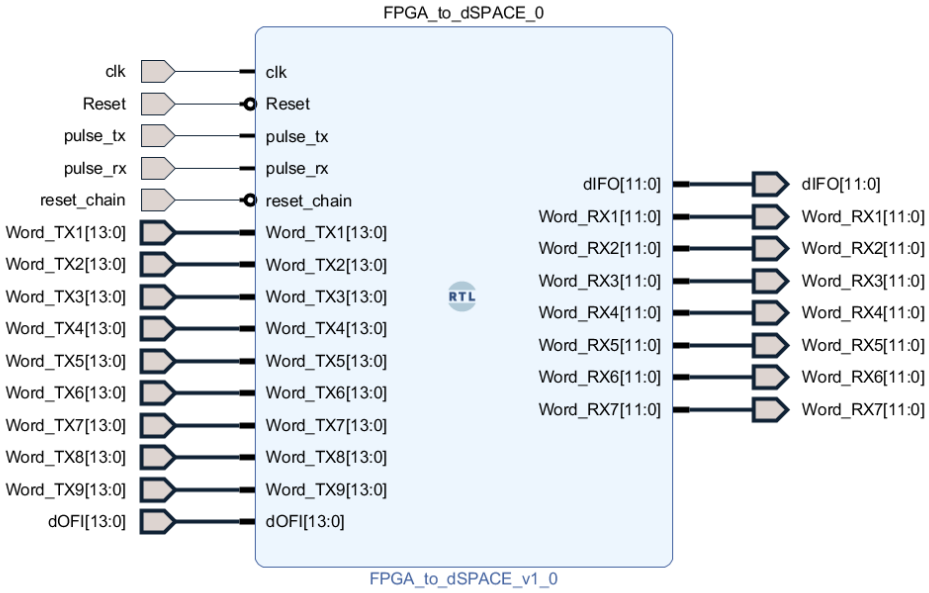


Figura 52 - Modulo di comunicazione con dSPACE.

In Figura 52 è possibile osservare una serie di segnali, chiamati $Word\_RX$ e $Word\_TX$, in formato *std logic* che rappresentano rispettivamente i dati in ingresso contenuti nel DOFI, opportunamente elaborati, e i dati in uscita dal sistema che devono essere elaborati e caricati in sequenza sul vettore uscente DIFO. In particolare, il modulo riceve in ingresso i dati in uscita dal blocco di gestione ADC, contenenti le acquisizioni analogiche provenienti dalla scheda di acquisizione CAB, e il registro errori proveniente dal modulo di protezioni HW. Tali dati vengono caricati in sequenza sul vettore DIFO in uscita dal modulo, grazie al seguente codice.

```
if(pulse_tx='1')then
 case conunt_tx_i is

 when 1 => dIFO<=Word_TX2;
    conunt_tx_i<=2;

 when 2 => dIFO<=Word_TX3;
    conunt_tx_i<=3;

 when 3 => dIFO<=Word_TX4;
    conunt_tx_i<=4;

 when 4 => dIFO<=Word_TX5;
    conunt_tx_i<=5;

 when 5 => dIFO<=Word_TX6;
    conunt_tx_i<=6;

 when 6 => dIFO<=Word_TX7;
    conunt_tx_i<=7;

 when 7 => dIFO<=Word_TX8;
    conunt_tx_i<=8;

 when 8 => dIFO<=Word_TX9;
```

```
    conunt_tx_i<=1;
 when others => conunt_tx_i<=1;
 end case;
end if;
```

Dal vettore DOFI vengono invece estratti i segnali utili al controllo implementato su FPGA. Si tratta dei duty cycle calcolati dal controllo di alto livello per i due inverter che, come detto nel paragrafo dedicato, vengono poi mandati in ingresso al modulatore PWM. Il vettore in ingresso contiene inoltre il registro stati che viene successivamente inviato al modulo FSM per la decodifica. Il codice di estrapolazione è il seguente.

```
if(pulse_rx='1')then
  case conunt_rx_i is

   when 1 => Word_RX1<=dOFI;
      conunt_rx_i<=2;

   when 2 => Word_RX2<=dOFI;
      conunt_rx_i<=3;

   when 3 => Word_RX3<=dOFI;
      conunt_rx_i<=4;

   when 4 => Word_RX4<=dOFI;
      conunt_rx_i<=5;

   when 5 => Word_RX5<=dOFI;
      conunt_rx_i<=6;

   when 6 => Word_RX6<=dOFI;
      conunt_rx_i<=7;

   when 7 => Word_RX7<=dOFI;
      conunt_rx_i<=1;

   when others => conunt_rx_i<=1;
  end case;
 end if;
```

Sia per la scrittura che per la lettura dei dati si nota fra le righe di codice mostrate la necessità di due segnali, ovvero $pulse\_tx$ e $pulse\_rx$. Questi indicano rispettivamente i comandi di inizio scrittura e lettura di ogni word. Al modulo FPGA_to_dSPACE viene quindi abbinato un blocco di gestione che, oltre a generare tali impulsi, permette la sincronizzazione fra i due sistemi comunicanti. Il modulo IRQ_Manager ha, infatti, l'importante compito di generare il segnale IRQ_dSPACE, ovvero l'interrupt che viene mandato a dSPACE per sincronizzare tutte le operazioni di comunicazione. Poiché i dati verso dSPACE riguardano le acquisizioni analogiche, si sottolinea che tramite il segnale IRQ generato dal modulo di gestione si effettua la sincronizzazione anche con il blocco ADC di gestione delle acquisizioni analogiche.

Tale modulo è mostrato in Figura 53.

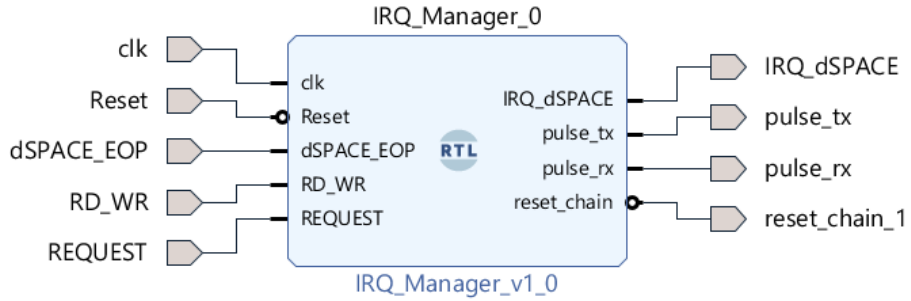


Figura 53 - Modulo IRQ Manager.

Oltre ai segnali provenienti dal System_Manager di Reset e clock per la temporizzazione, il blocco riceve in ingresso da dSPACE tre segnali per una corretta comunicazione. $dSPACE\_EOP$ (End of Operation) viene tirato giù ad ogni interrupt e rimane basso per tutta la durata della comunicazione, per essere riportato alto quando il master, ovvero dSPACE, segnala la fine del processo. Durante questo periodo il segnale RD_WR viene portato ad 1 o a 0 per indicare rispettivamente le operazioni di lettura o scrittura e viene lasciato allo stato basso alla fine di ogni comunicazione. Infine, il segnale $REQUEST$ è composto da una serie di impulsi corrispondenti all'inizio dell'operazione di trasmissione di ciascuna parola.

Implementazione e simulazione del sistema con FPGA_to_dSPACE

Di seguito si riportano i risultati ottenuti con una simulazione di tipo comportamentale, nella quale è possibile osservare la sequenza dei segnali di gestione della comunicazione.

Il segnale IRQ, arancione in Figura 54 è l'interrupt generato per essere mandato a dSPACE. I tre segnali successivi vengono generati da testbench seguendo la stessa logica che verrà implementata su dSPACE per la generazione degli stessi.

Si osservi che in corrispondenza dei fronti discendenti del segnale $REQUEST$ vengono generati gli impulsi di lettura e scrittura dei dati, rispettivamente $pulse\_tx$ e $pulse\_rx$.

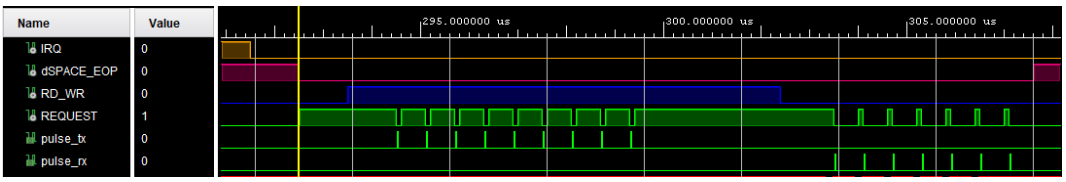


Figura 54 - Behavioral simulation del sistema con FPGA_to_dSPACE.

Si osserva infatti che successivamente al fronte di discesa del segnale di interrupt, dopo un certo ritardo anche $dSPACE\_EOP$ passa allo stato logico 0. Iniziano quindi le operazioni di lettura da dSPACE, nelle quali le nove parole in uscita vengono trasmesse dall'FPGA, una ad ogni colpo di $REQUEST$. Quando poi il RD_WR viene portato alto, inizia allora la scrittura da parte di dSPACE dei sette vettori di dati necessari all'FPGA.

2.3.10 Ventilatore di raffreddamento

A causa delle perdite per conduzione e commutazione dell'inverter, e della presenza di resistenze fisiche e parassite nel sistema, questo disperde una certa quantità di energia termica, che deve essere opportunamente smaltita. A questo scopo gli stack Semikron [11] utilizzati nel corso della presente tesi sono provvisti di un sistema di raffreddamento con ventilatori centrifughi collegati a dissipatori alettati posti sul fondo degli stack di potenza, come anticipato al Paragrafo 1.1.

Il ventilatore, di cui si riporta il riferimento al datasheet in bibliografia [13], presenta la possibilità di una regolazione del suo periodo di funzionamento, che richiede che gli vengo fornito un segnale di comando generato con un modulatore PWM. Si implementa allora su Vivado il modulo di gestione del ventilatore, rappresentato in Figura 55.

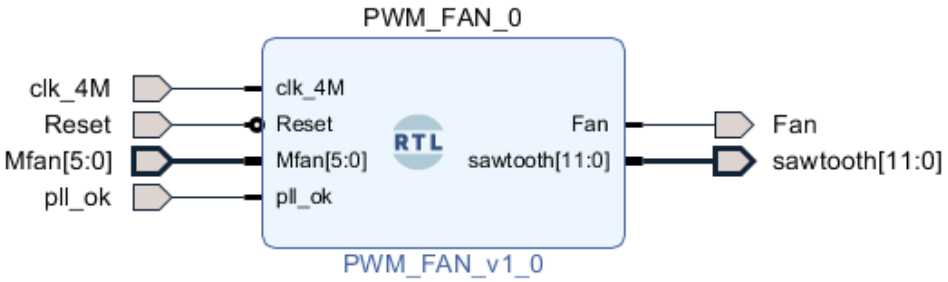


Figura 55 - Modulo PWM_Fan.

Il modulo riceve in ingresso il segnale di $Reset$ del sistema, un clock a 4 MHz appositamente generato e l'indice di modulazione, in arrivo da dSPACE. L'indice di modulazione viene quindi confrontato con un segnale portante, per produrre in uscita il comando da mandare al ventilatore. Di seguito si riporta il codice sviluppato per il funzionamento del modulo descritto.

```
if(rising_edge(clk_4M)) then
          if (Reset='0')then
           if (counter = Tr_up) then
          counter <= zeroT;
          else
           counter <= counter + oneT;
           end if;

                if(counter < unsigned(M_fan))then
                    fan_sig <= '1';
                else
                    fan_sig <= '0';
                end if;
        else
```

```
            fan_sig <= '0';
            counter <= zeroT;
        end if;
    end if;
```

Implementazione e simulazione del PWM_Fan

Grazie alla definizione di un counter si genera, internamente al modulo, un segnale portante a dente di sega alla frequenza di 1 kHz, con il quale confrontare l'indice di modulazione in formato std_logic_vector inviato da dSPACE. In Figura 56 si può osservare la logica implementata, secondo la quale il comando del ventilatore viene portato ad 1 ogni volta che l'indice di modulazione diventa maggiore del segnale portante.

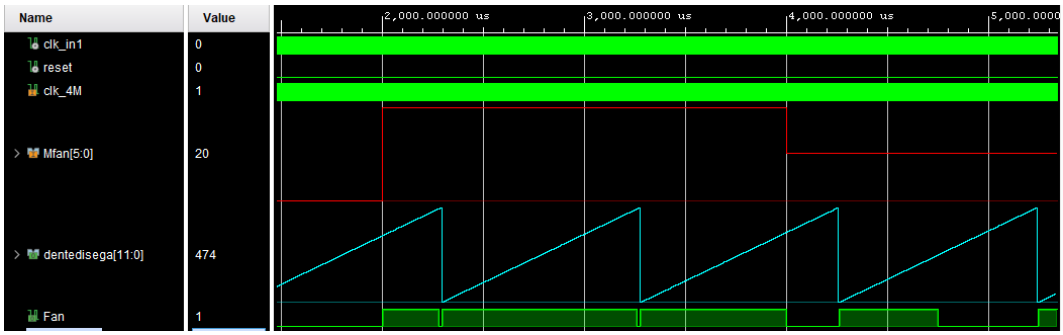


Figura 56 - Behavioral simulation del PWM_Fan.

2.3.11 Led di stato

Si implementa infine un modulo per la gestione di alcuni led montati a bordo della scheda FPGA, che si mostra in Figura 57.

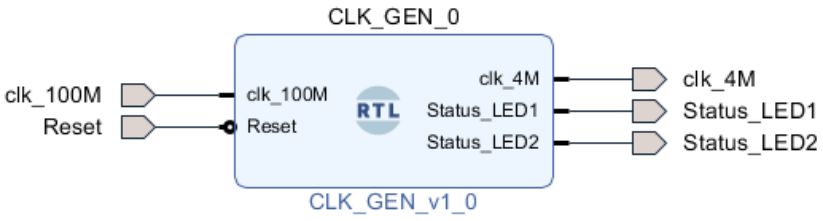


Figura 57 - Modulo CLK_GEN.

Tale modulo consente, tramite l'accensione di due led, di verificare visivamente il caricamento del codice implementato sulla scheda FPGA e l'eventuale presenza del segnale di reset del sistema. Tramite il codice di seguito riportato si comanda infatti un led con un clock a bassa frequenza, la cui accensione periodica indica il corretto caricamento del codice sulla scheda, e un altro led con il segnale di reset della scheda, così che la sua attivazione possa essere tempestivamente identificata.

```
if(rising_edge(clk_100M)) then
```

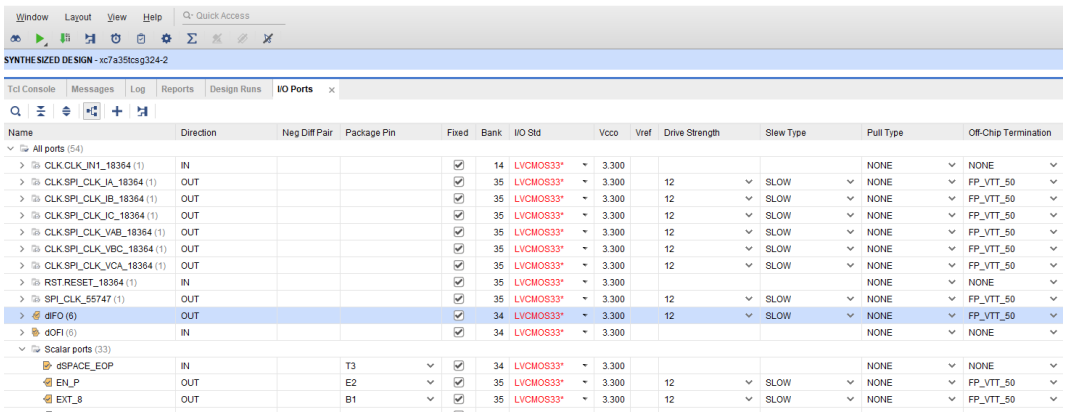
```
            if (Reset='0')then
                if(counterLED = LED_DIV-1) then
                    counterLED := 0;
                    clk_LED := NOT clk_LED;
                else
                    counterLED := counterLED + 1;
                end if;
            else
                counterLED := 0;
                clk_LED := '0';

            end if;
        end if;
        LED <= clk_LED;

    Status_LED1 <= LED;
    Status_LED2 <= not Reset;
```

2.4 Programmazione non volatile dell'FPGA

Una volta che il codice VHDL scritto risulta soddisfacente è possibile avviare il processo di sintesi, implementazione e generazione del bitstream. Queste fasi corrispondono rispettivamente alla generazione della *Netlist*, al piazzamento del circuito in FPGA e alla generazione del Bitstream da cariare direttamente sulla scheda di controllo. Il processo di sintesi traduce, infatti, il codice VHDL in una descrizione a livello di porte logiche (o *Netlist*). In questa fase si assegnano gli ingressi e le uscite dei moduli ai pin fisici presenti sulla scheda, configurandone la tipologia, ad esempio, LVCMOS33. In Figura 58 si mostra un esempio di assegnazione dei pin nella sezione I/O Ports di Window.



Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination
All ports (54)												
CLK.CLK_IN1_18364 (1)	IN			☑	14	LVCMOS33*	3.300				NONE	NONE
CLK.SPI_CLK_IA_18364 (1)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
CLK.SPI_CLK_IB_18364 (1)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
CLK.SPI_CLK_IC_18364 (1)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
CLK.SPI_CLK_VAB_18364 (1)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
CLK.SPI_CLK_VBC_18364 (1)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
CLK.SPI_CLK_VCA_18364 (1)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
RST.RESET_18364 (1)	IN			☑	35	LVCMOS33*	3.300				NONE	NONE
SPI_CLK_55747 (1)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
dIFO (6)	OUT			☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
dOFI (6)	IN			☑	34	LVCMOS33*	3.300				NONE	NONE
Scalar ports (33)												
dSPACE_EOP	IN		T3	☑	34	LVCMOS33*	3.300				NONE	NONE
EN_P	OUT		E2	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
EXT_8	OUT		B1	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50

Figura 58 - I/O Ports del progetto Vivado sintetizzato.

Generato il Bitstream, questo può essere utilizzato per programmare la scheda tramite un apposito programmatore che manda i dati alla scheda attraverso il connettore JTAG di Figura 59.

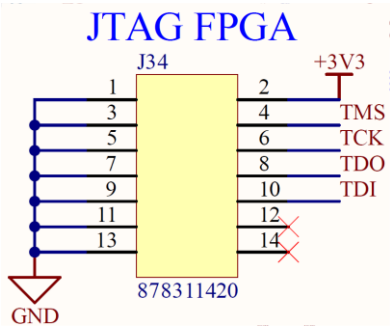


Figura 59 - Connettore JTAG per la programmazione dell'FPGA.

Si sottolinea ora che le FPGA sono volatili, quindi non mantengono la programmazione, perciò bisogna usare supporti di memorizzazione esterni e generare il file di configurazione adatti. La scheda FPGA è, infatti, dotata di una memoria esterna, 5 in Figura 60, di tipo flash QSPI integrata da 32 MByte S25FL256S [2], che viene utilizzata per memorizzare la configurazione FPGA iniziale. Oltre alla configurazione FPGA, è possibile utilizzare la memoria flash rimanente per l'applicazione utente, ovvero è possibile caricare su questa il Bitstream generato in Vivado, in modo da non perdere la memorizzazione del codice ogni volta che la scheda viene disalimentata.

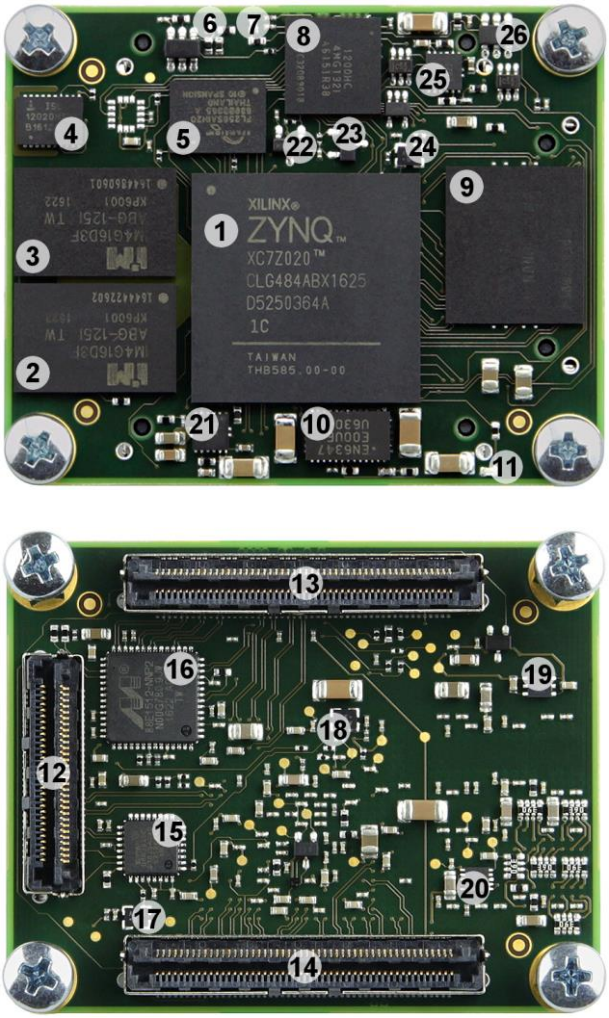


Figura 60 - Componenti principali del modulo TE0720 [6].

Su Vivado si configura allora il dispositivo di memoria e si genera l'opportuno file.mcs a partire dal Bitstream generato. Nella sezione Hardware manager si seleziona il dispositivo di memoria desiderato, che viene dunque programmato e verificato. Si procede, infine, al Booting della scheda FPGA.

In Figura 61 si mostra il diagramma di flusso delle operazioni da eseguire allo scopo di caricare il codice implementato sulla scheda FPGA, utilizzando la sua memoria volatile per una programmazione provvisoria o la memoria flash se si desidera conservare la programmazione anche in caso di disalimentazione della scheda stessa.

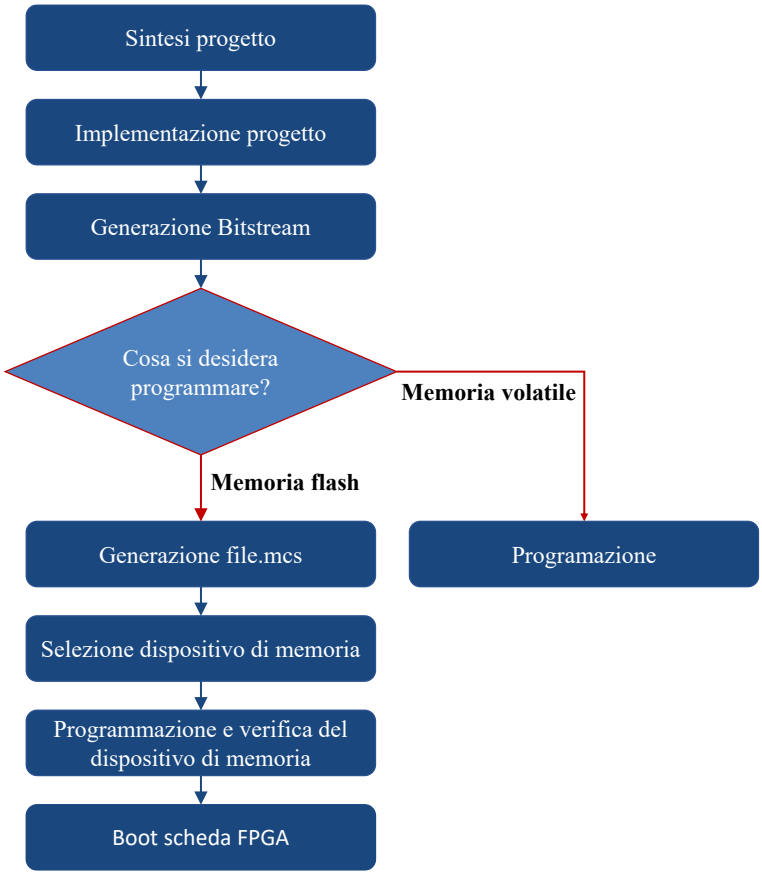


Figura 61 - Diagramma di flusso.

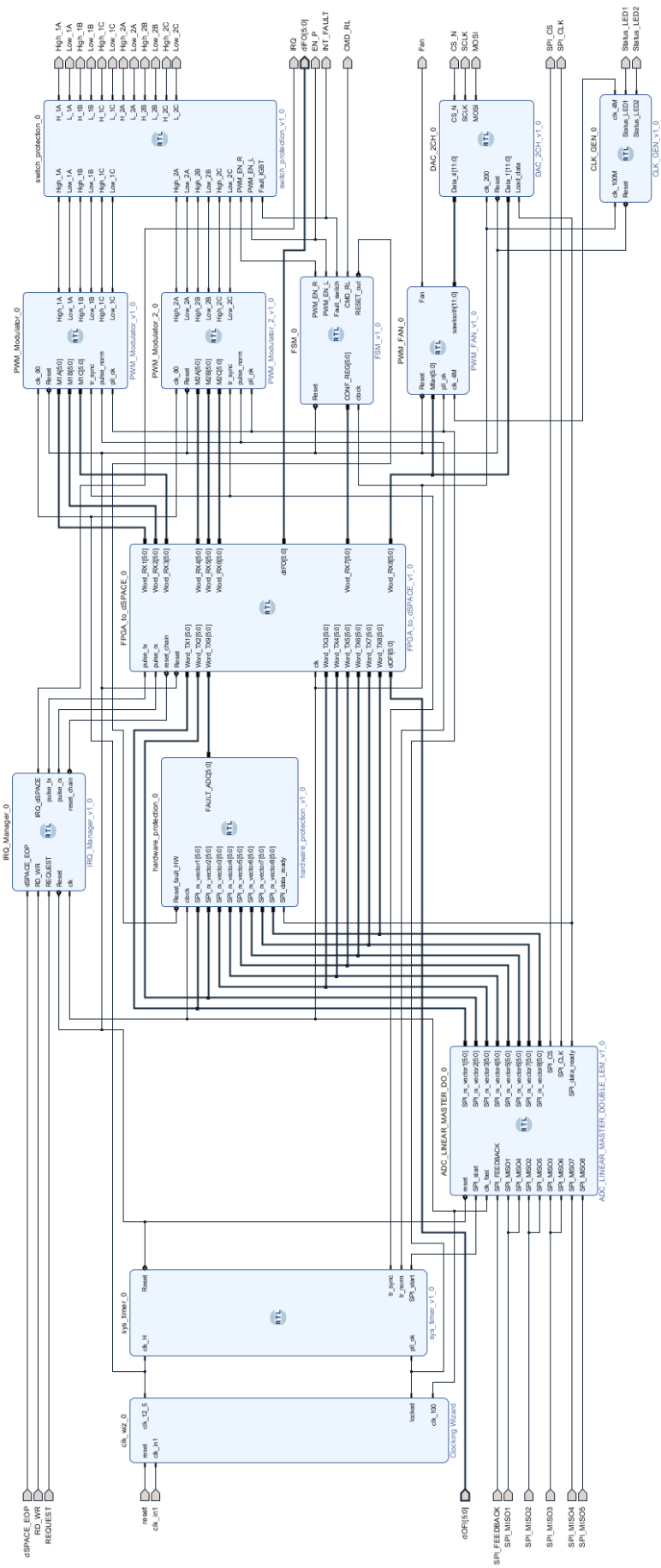


Figura 62 - Sistema Vivado completo.

Capitolo 3

Power Hardware in the Loop per testing convertitori di potenza

Con hardware in the loop (HIL) si indicano le tecniche di verifica di unità di controllo elettronico ad appositi banchi che riproducono in modo più o meno completo il sistema elettrico ed elettronico dell'applicazione a cui sono destinate. Il PHIL offre quindi la possibilità di caratterizzare il comportamento dell'hardware.

Lo scopo ultimo di questo lavoro di tesi è quello, appunto, di realizzare un banco di prova che consenta di testare i diodi presenti nei moduli di potenza dei due stack Semikron che si sono presentati al Capitolo 1. Per fare ciò si implementa in dSPACE il controllo di corrente e di tensione ad alto livello del sistema, che viene però precedentemente realizzato e testato in simulazione con l'ausilio del software di simulazione PLECS. Il sistema finale risulta infatti costituito dai due power stack che si interfacciano con dSpace e con la scheda FPGA, responsabile del controllo di basso livello, per cui si riporta al Capitolo 2 dedicato.

Di seguito si presenta lo schema completo del sistema che si intende realizzare in cui è possibile distinguere in alto i due moduli a IGBT, alimentati da una stessa sorgente DC e collegati fra loro tramite un filtro LCL. In basso invece sono presenti il blocco col codice di controllo relativo a dSPACE e i blocchi modulatore PWM e ADC, implementati in Vivado per la scheda FPGA.

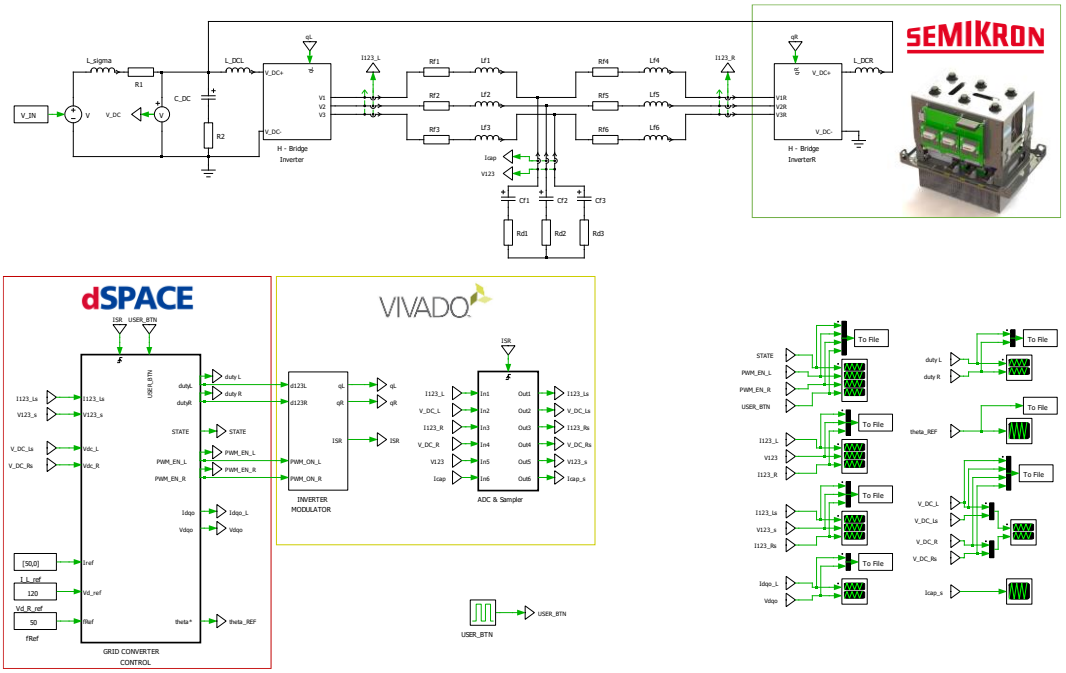


Figura 63 - Schema elettrico e di controllo del modello PLECS.

Per il collaudo finale del convertitore di potenza si collegheranno i due stack gemelli Semikron alla medesima sorgente di tensione continua, disaccoppiandoli tramite un filtro LC.

Le due unità vengono di seguito contrassegnate tramite le lettere L, che indica l'unità di sinistra, e R, per l'unità di destra. Di queste l'unità L rappresenta il convertitore da testare (CUT – Converter Under Test) che verrà controllato in corrente con un regolatore PI, mentre sull'unità di destra si attua un controllo di tensione in anello aperto. Tale setup permette di esplorare qualsiasi punto di funzionamento dello stack L, variando lo sfasamento fra il vettore rotante delle tensioni dell'unità R e quello delle correnti del convertitore L.

In Figura 64 si riporta lo schema di principio dei due anelli di controllo. Con riferimento alla Figura si osserva che quello che si intende implementare è un controllo di tipo vettoriale in assi dqo, le cui grandezze verranno da ora contrassegnate per semplicità con le lettere maiuscole, mentre con le minuscole si indicheranno le grandezze in assi 123. Al Paragrafo 3.2.1 si discuterà nel dettaglio il processo di implementazione del controllo e si introdurranno alcune possibili migliorie giustificando la scelta finale del loro utilizzo.

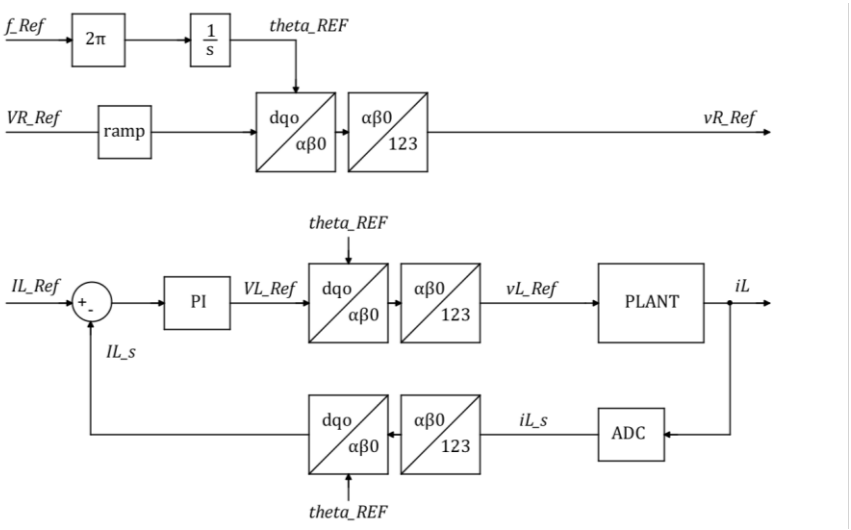


Figura 64 - Schema di principio degli anelli di controllo.

In Figura 65 è riportato lo schema elettrico inverter trifase ad IGBT con diodi, nel quale è rappresentato anche il condensatore di DC-link di capacità pari a $7{,}05mF$, come da datasheet.

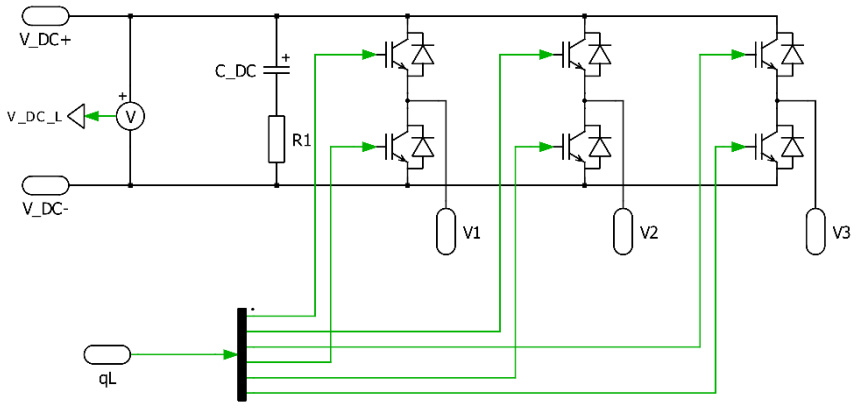


Figura 65 - Schema dell'inverter trifase a IGBT con relativa capacità di DC-link.

I comandi di gate degli switch alti e bassi, indicati come qL e qR, sono prodotti dal blocco modulatore PWM, implementato in Vivado per la scheda FPGA, che riceve a sua volta i duty cycle da dSPACE, come mostrato in Figura 63.

3.1 ADC e Modulatore PWM

Ai fini del controllo è necessaria l'acquisizione di alcune grandezze, per cui si modellizza in PLECS il blocco di conversione dei segnali da analogici a digitali. Questo viene rappresentato in maniera ideale, cioè senza guadagni né discretizzazioni, il che costituisce un'approssimazione accettabile poiché gli ADC del sistema reale, avendo 14 bit di risoluzione, sono molto performanti.

In Figura 66 si riporta lo schema del blocco ADC, dove le grandezze campionate vengono contrassegnate con il pedice '*s*'.

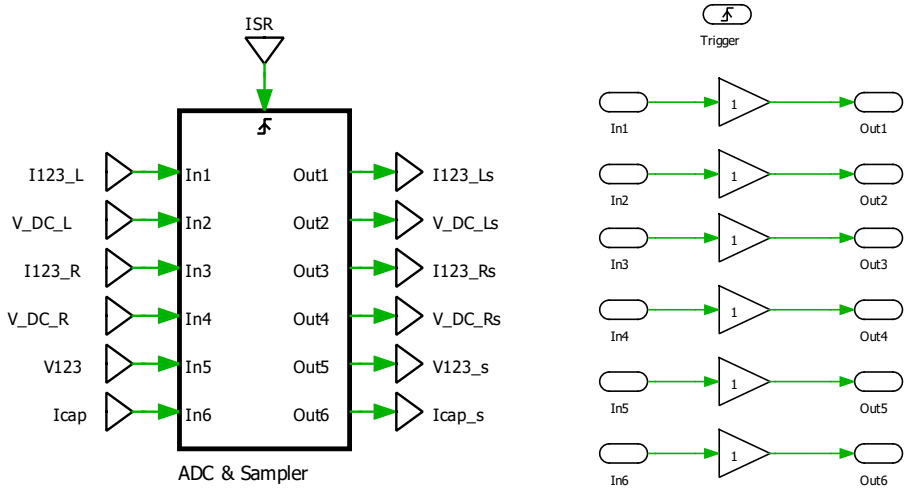


Figura 66 - Blocco ADC.

Si implementa poi il blocco modulatore PWM, che riceve dal controllo i duty cycle e i segnali di abilitazione della modulazione e restituisce in uscita i comandi degli switch degli inverter. In Figura 67 si possono distinguere i generatori di triangola di modulo unitario a frequenza di commutazione. Queste vengono comparate con gli ingressi ad ogni colpo di Interrupt per generare le uscite che vengono mandate direttamente all'inverter.

Si noti infine che il segnale ISR di Interrupt viene mandato anche al modulo ADC precedentemente descritto e al blocco di controllo, in modo da sincronizzare le operazioni di acquisizione, modulazione e controllo.

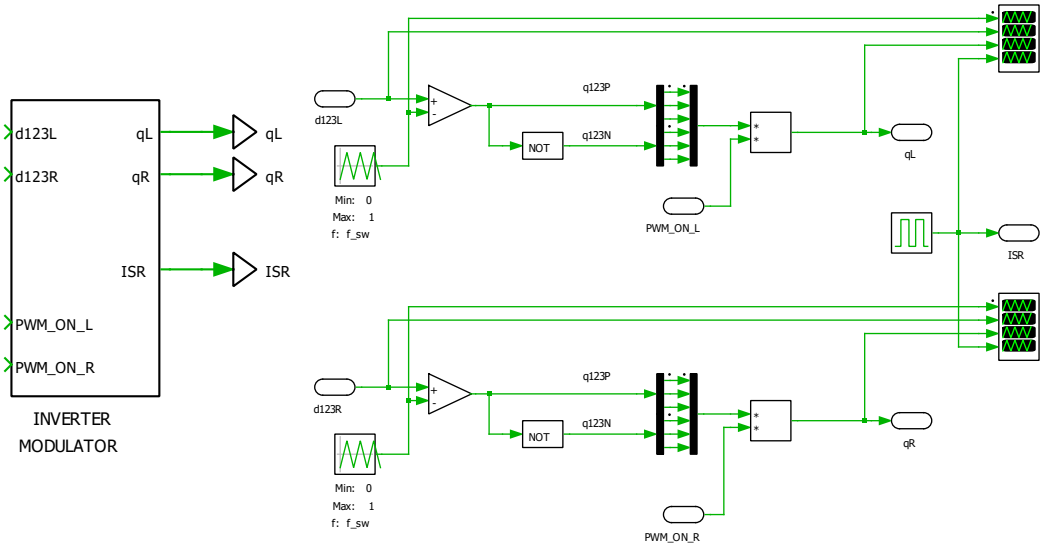


Figura 67 - Blocco Modulatore PWM.

3.2 Blocco di controllo C-Script

Si tratta di un blocco scritto in linguaggio di programmazione C che riceve in ingresso le grandezze campionate ed i riferimenti ed esegue il controllo ad alto livello per regolare tensioni e correnti dei due convertitori.

Questo riceve in ingresso il segnale ISR, precedentemente esposto, e il segnale USER_BTN, che rappresenta lo start in arrivo da dSPACE. In uscita si hanno quindi i duty cycle dei due inverter e alcune variabili di debug aggiuntive.

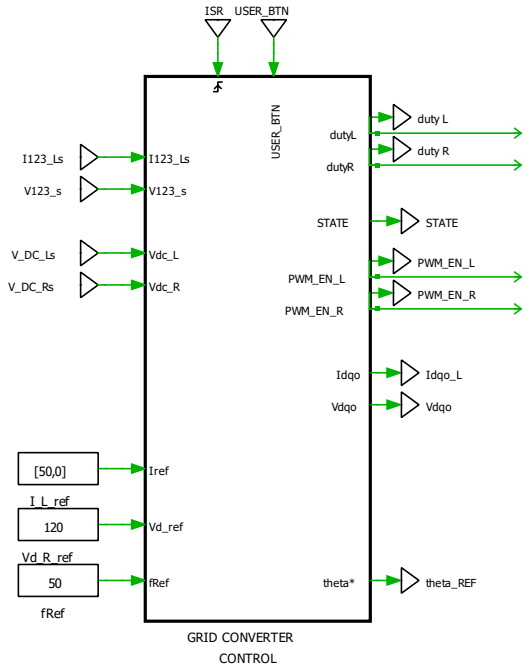


Figura 68 - Blocco di controllo.

Internamente il blocco si mostra come in Figura 69, con gli ingressi a sinistra e le uscite a destra, dove è possibile notare dei blocchi Z^{-1} che modellizzano il ritardo di un passo di calcolo del controllo digitale.

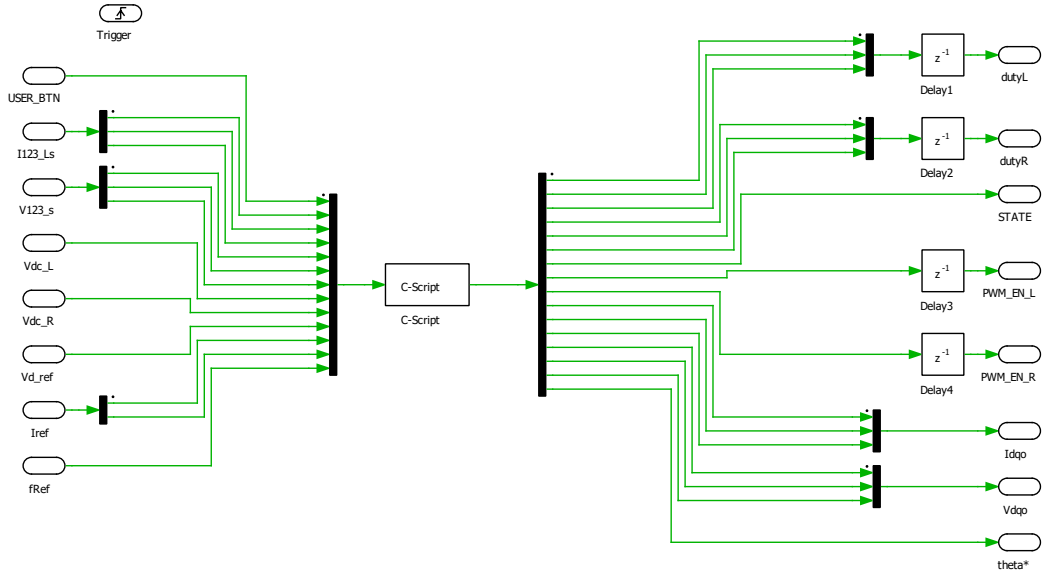


Figura 69 - Blocco C-Script.

Il blocco C-Script è suddiviso in tre sezioni:

- Declaration, dove si dichiarano le librerie contenenti costanti, tipi di dati, variabili utente e routine di controllo.
- Start Function, cioè un codice di inizializzazione che viene chiamato solo una volta all'inizio della simulazione. Qui si trova l'inizializzazione degli stati e dei parametri del controllo.
- Output Function, che è il codice di controllo richiamato ad ogni Interrupt. Questo inizia con la dichiarazione dei canali degli input e finisce con la dichiarazione degli output. Nel mezzo vi sono le stringhe di controllo comprendenti protezioni software, filtri, trasformazioni di coordinate, gestione degli stati e anelli di controllo.

3.2.1 Schema di controllo

Il controllo di alto livello, a carico di dSPACE, consiste nel controllo di corrente dell'unità L in anello chiuso e nel controllo di tensione dello stack R in anello aperto.

Supponendo che la tensione sui condensatori lato AC possa considerarsi costante, ai fini del dimensionamento del regolatore dell'anello di corrente è possibile approssimare il sistema ad un carico puramente induttivo e ricavare la relazione fra la tensione in ingresso al convertitore e la corrente in uscita, ovvero la grandezza da controllare. Infatti, il sistema in questione è rappresentato da carico RL con R trascurabile collegato tra due generatori di tensione, ovvero l'inverter e i condensatori.

Nel dominio di Laplace risulta:

$$i\_L_{123} = v\_L_{123} \cdot \frac{1}{sL}$$

Il controllo di corrente viene eseguito con regolatore PI, nel quale si imposta un guadagno Kp tale da ottenere una banda passante a frequenza pari a $1/20$ della frequenza di commutazione.

$$\mathrm{Kp} = 2\pi \cdot f_{BW} \cdot L$$

Dove:

$$f_{BW} = \frac{1}{20} f_{SW} = 500\ Hz$$

Per quanto riguarda il guadagno Ki della parte integrale, questo si sceglie in maniera da imporre lo zero del regolatore ad un decimo della pulsazione della banda passante.

$$\omega_z = \frac{\mathrm{Ki}}{\mathrm{Kp}} = \frac{1}{10}\omega_{BW}$$

$$\mathrm{Ki} = \frac{1}{10} \cdot 2\pi \cdot f_{BW} \cdot \mathrm{Kp}$$

Di seguito si riporta lo schema a blocchi nel dominio di Laplace dell'anello di corrente descritto.

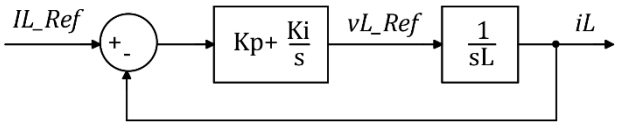


Figura 70 - Controllo di corrente con PI in anello chiuso.

Si esegue un controllo vettoriale in assi dqo, per cui nei due anelli di controllo si effettuano delle trasformazioni di assi di riferimento che permettono di confrontare le grandezze misurate in assi 123, con quelle di riferimento, date in assi dqo. Tali trasformazioni sono le ben note trasformazioni di Park e Clarke, di cui la seconda necessita di un riferimento d'angolo per la rotazione. L'angolo di riferimento viene dunque generato internamente per integrazione, a partire da una frequenza di riferimento posta pari a $50\ Hz$, come mostrato nella figura seguente.

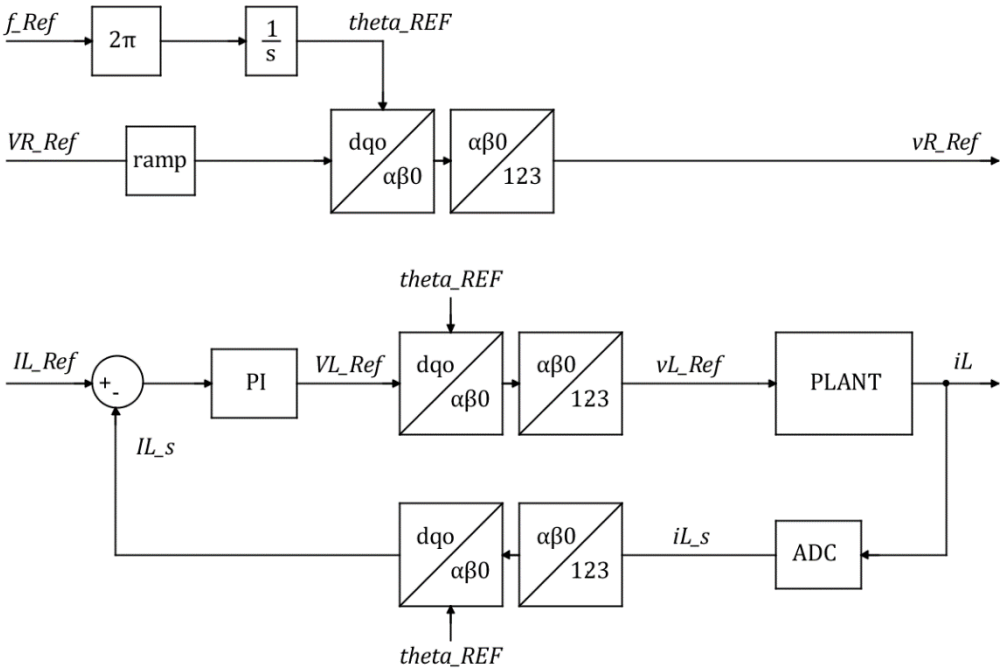


Figura 71 - Schema di controllo di i_L e v_R.

Per semplicità si indicano con le lettere maiuscole le grandezze in assi dqo e con le lettere minuscole le grandezze in assi abc.

3.2.2 Rampa di tensione – effetto del Feed-Forward

Si inserisce nell'anello di controllo di corrente un Feed-Forward di tensione al fine di migliorarne la dinamica.

Chiamando v_L_{123} la tensione in uscita dal convertitore di sinistra e v_R_{123} la tensione in ingresso all'unità di destra risulta:

$$v\_L_{123} = L \cdot \frac{di}{dt} + v\_R_{123}$$

Questa tecnica di controllo consente infatti di ridurre l'errore già dai primi istanti di funzionamento, evitando così i picchi iniziali di corrente.

Si sceglie inoltre di far variare la tensione del convertitore R a rampa e solo quando la rampa è completa si abilita il controllo di corrente. Tale procedura di avviamento è stata validata in simulazione, come si vede in Figura 72. Nei primi $25\,ms$ la tensione di riferimento viene variata a rampa, in modo da evitare sovracorrenti di inserzione sui condensatori. Una volta che la tensione si è stabilizzata, il controllo di corrente dell'unità di sinistra viene abilitato ($t = 140ms$). Si nota infine una leggera variazione nella tensione dei condensatori, legata al transitorio di avviamento del controllo di corrente.

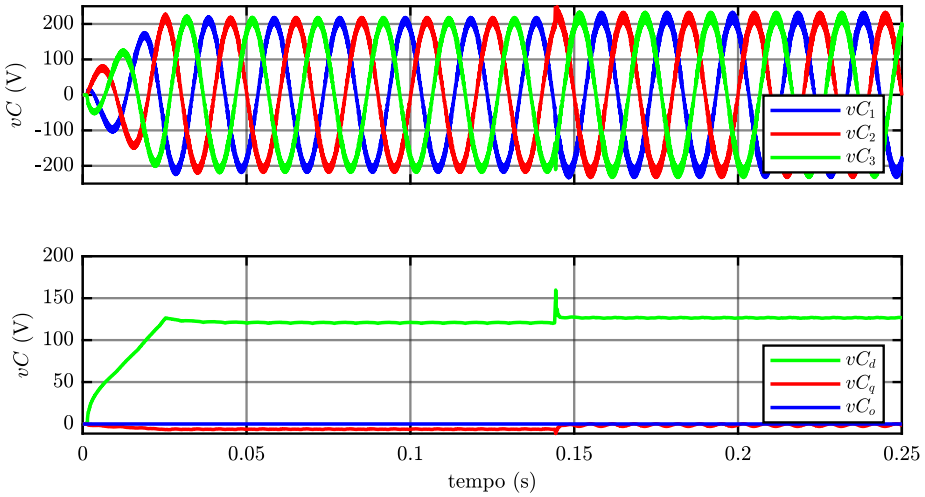


Figura 72 - Tensione controllata in assi 123 in alto e dqo in basso.

3.2.3 Saturazione del regolatore di corrente

Si opta per un tipo saturazione del regolatore che privilegia l'asse d, poiché è in questo che si trasmette la potenza attiva, in quanto allineato con la tensione di riferimento dei condensatori. Si limita allora la tensione media di uscita del convertitore in asse d al massimo valore possibile e si ricavano i limiti sugli altri due assi per differenza, misurata e nota la tensione del DC-link V_L_DC.

$$ILd_{lim} = \frac{V\_L\_DC}{\sqrt{3}}$$

$$ILq_{lim} = ILo_{lim} = \sqrt{\frac{V\_L\_DC^2}{\sqrt{3}} - V\_Ld_{REF}^2}$$

3.2.4 Regolatore Risonante PRes

In Figura 73 si nota che nonostante le componenti d e q di corrente seguano perfettamente i loro riferimenti è presente una componente omopolare non nulla che distorce la corrente risultante in assi 123.

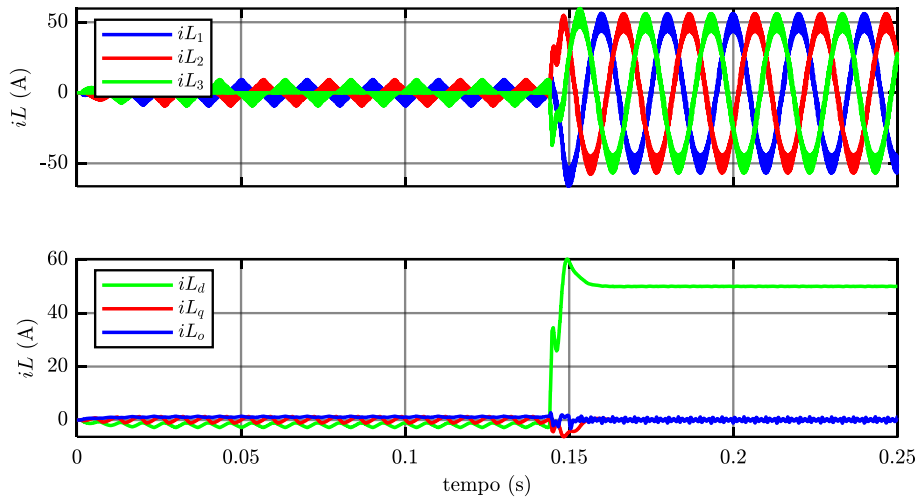


Figura 73 - Risposta a gradino della corrente I_L in assi 123 in alto e dqo in basso.

Al fine di attenuare tale distorsione si inseriscono, in parallelo al regolatore PI dell'anello in asse o, due regolatori risonanti atti ad eliminare la terza e nona armonica, che in accordo con l'analisi armonica di Figura 74 si ritengono essere le uniche significative.

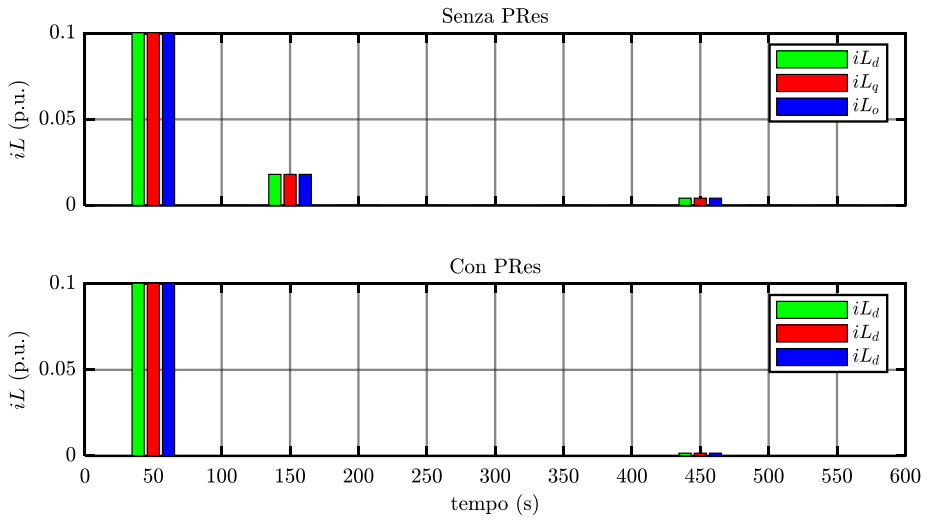


Figura 74 - FTT della corrente I_L in assi dqo.

Tali regolatori sono caratterizzati da guadagno nullo su tutto lo spettro ad eccezione della frequenza di risonanza imposta, in corrispondenza della quale hanno guadagno idealmente infinito.

La loro FDT, di cui si riporta il diagramma di Bode in Figura 75, è la seguente:

$$RES(s) = KRes \cdot \frac{s}{s^2 + \omega_0^2}$$

Dove ω_0 è la pulsazione corrispondente alla frequenza di risonanza.

$$\omega_0 = 2\pi \cdot f_{RES}$$

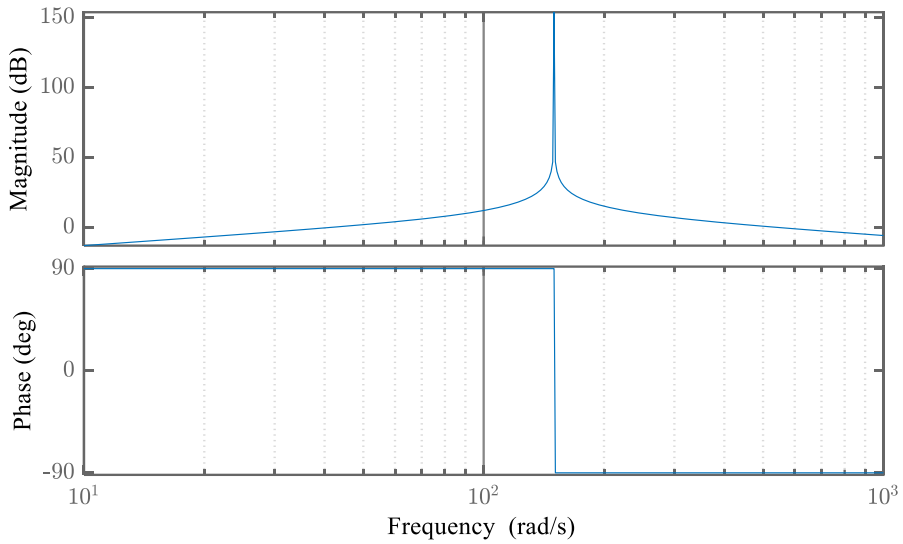


Figura 75 - Diagramma di Bode di un regolatore PRes, $\omega_0 = 150$ *Hz*.

Al fine di eliminare la terza e nona armonica si pongono dunque i riferimenti di tali armoniche a 0 e le frequenze di risonanza dei due regolatori pari a:

$$f_{RES3} = 3 \cdot \omega_{\mathrm{REF}} \qquad f_{RES9} = 9 \cdot \omega_{REF}$$

La FDT precedentemente scritta è valida però solo nel dominio continuo del tempo. Per portare tale formulazione nel discreto ed adattarla alle necessità del controllo digitale è necessario applicare agli ingressi x_1 e x_2, che sono le variabili di stato, la seguente matrice di coefficienti, calcolati in funzione del guadagno $KRes$, della frequenza di risonanza ω_0 e del passo di discretizzazione T_s.

$$A = \begin{bmatrix} \cos(\omega_0 T_s) & \sin(\omega_0 T_s) \\ -\sin(\omega_0 T_s) & \cos(\omega_0 T_s) \end{bmatrix}$$

$$B = \frac{2 \cdot KRes}{\omega_0} \begin{bmatrix} sin(\omega_0 T_s) \\ cos(\omega_0 T_s) - 1 \end{bmatrix}$$

Tali coefficienti sono adattivi, in quanto dipendono dalla frequenza di riferimento dell'unità R, e vengono perciò inseriti nella sezione *Output Function* del blocco C-Script.

Lo schema a blocchi risultante del comparatore dell'anello di corrente in assi dqo è mostrato nella figura seguente.

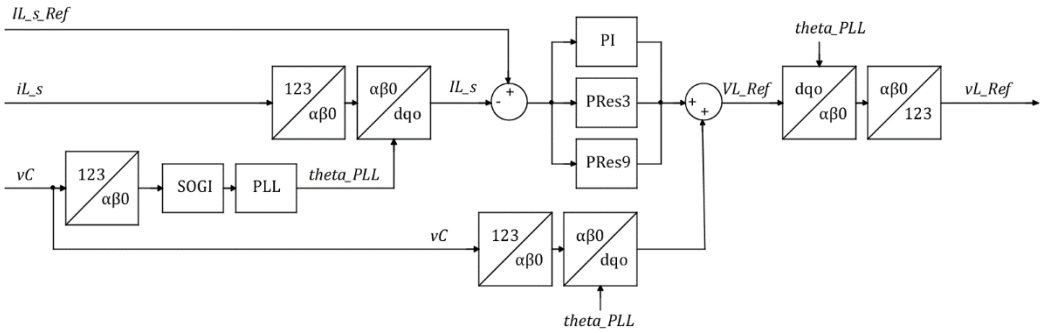


Figura 76 - Schema a blocchi dell'anello di corrente in assi dqo.

Si mostrano infine i risultati ottenuti inserendo i regolatori risonanti nell'anello di controllo di corrente. In Figura 77 si osserva come la corrente in asso o risulti notevolmente meno distorta rispetto al caso precedente presentato in Figura 73.

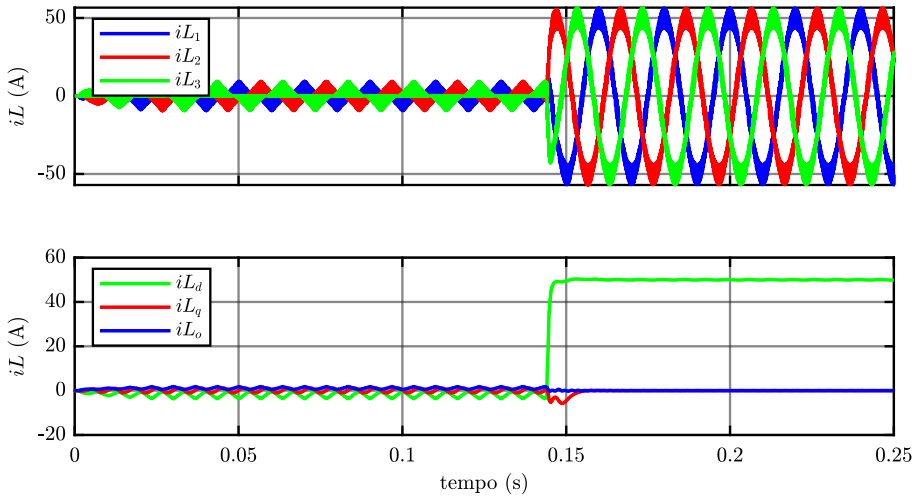


Figura 77 - Corrente I_L in assi abc in alto e dqo in basso, con PRes.

Si coglie inoltre l'occasione per osservare la presenza di corrente sulle fasi prima dell'abilitazione della modulazione al tempo $t = 0{,}14\ s$. Questa è dovuta all'interfacciarsi dei due convertitori, infatti, l'unità R inietta nel sistema della corrente di modo comune che l'inverter L non è ancora in grado di controllare.

3.2.5 Effetto del disaccoppiamento del DC-link

Si inseriscono due filtri LC identici, uno per stack, di disaccoppiamento del DC-link per evitare disturbi reciproci causati dal ripple. I due componenti reattivi vengono dimensionati rispettando la seguente relazione che impone che la

frequenza di risonanza del filtro introdotto sia molto minore della frequenza di switching.

$$\frac{1}{2\pi \cdot \sqrt{L\_DC \cdot C\_DC}} \ll f_{SW}$$

Si sottolinea che la capacità C_DC in questione è quella interna allo stack, mostrata in Figura 65, che secondo datasheet vale:

$$C\_DC = 7{,}05\ mF$$

Quindi, ponendo la frequenza di risonanza del circuito LC in questione pari ad $1/10$ della frequenza di commutazione, l'induttanza risulta:

$$L\_DC = 0{,}36\ nF$$

Si noti che la capacità del condensatore è così grande che l'induttanza necessaria risulta piccola a sufficienza da non dover necessariamente rappresentare un componente aggiuntivo nel setup reale, ma può essere facilmente sostituita dall'induttanza degli stessi cavi di collegamento.

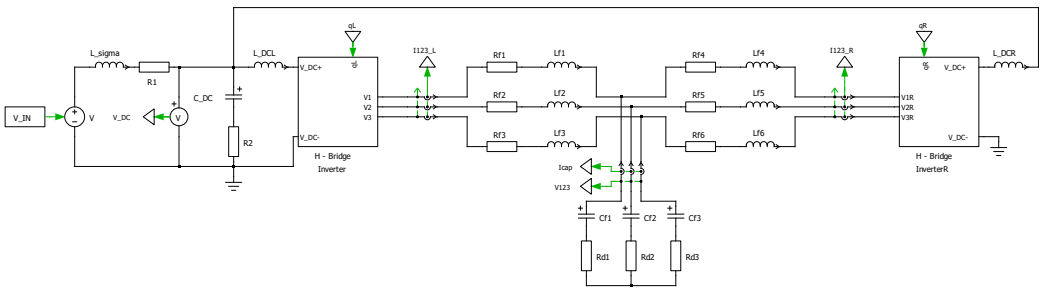


Figura 78 – Schematico con filtro LC.

In Figura 78 si osservano altri elementi reattivi quali *L_sigma*, che rappresenta l'induttanza dei cavi di collegamento, ed un secondo condensatore di capacità *C_DC*, esterno algli stack, che funge da sorgente di tensione ideale al fine di limitare gli effetti induttivi dei cavi sulla tensione di DC-link. Tali componenti vengono messi in serie con delle resistenze di piccolo valore che modellizzano i fenomeni reali di smorzamento causati dalle resistenze parassite.

Dopo aver disaccoppiato i DC-link si misurano le due tensioni VDC_R e VDC_L in ingresso ai due convertitori, che ora risultano diverse.

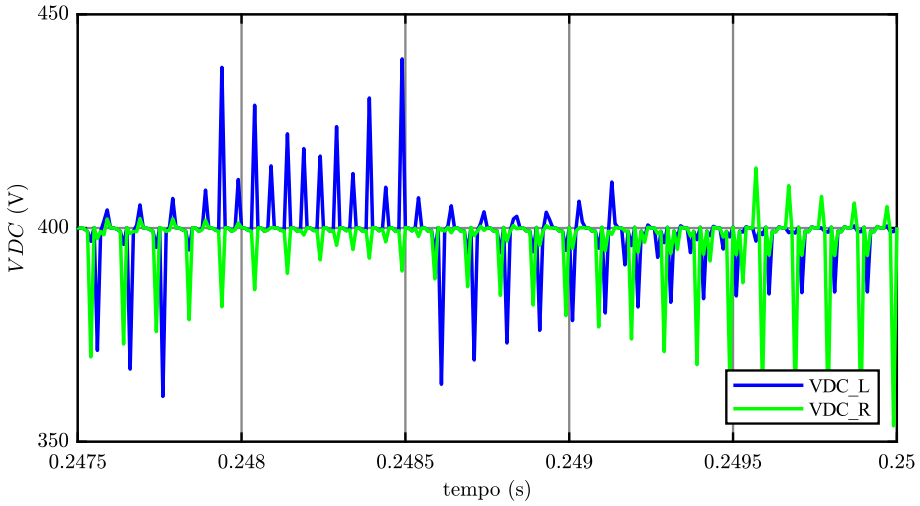


Figura 79 - Confronto VDC_R e VDC_L.

In Figura 79 è possibile notare la presenza di un ripple alla frequenza di commutazione su entrambe le forme d'onda, che risultano però efficacemente disaccoppiate. Si osserva che il ripple delle forme d'onda in Figura non è rappresentativo di un fenomeno reale, in quanto frutto di una simulazione con componenti ideali. L'inverter, così come è modellizzato in Figura 65, comprende degli IGBT ideali con comportamento analogo a dei generatori di corrente a gradino, ed un circuito RC. Le variazioni istantanee di corrente sugli IGBT generano dunque gli elevati ripple in Figura, il cui valore di picco è proporzionale alla resistenza e la cui legge di decadenza dipende dalla costante di tempo del suddetto circuito RC. Nel setup reale tale ripple alla frequenza di commutazione, seppur esistente, è notevole filtrato dai fenomeni reattivi parassiti.

Si provano inoltre diversi valori di sfasamento fra le portanti dei modulatori di ciascuna unità per indagare la forma del ripple sovrapposto al valor medio della corrente I_123 che circola fra i due stack. Si nota infine che poiché tale ripple è alla frequenza di commutazione, ovvero $10kHz$, sulle variabili di controllo non si manifesta alcun effetto. Queste vengono infatti campionate a metà del periodo di commutazione, avendo sincronizzato le acquisizioni dell'ADC al vertice della triangola del modulatore (Paragrafo 2.3.4), quindi se ne rileva solo il valor medio che rimane immutato.

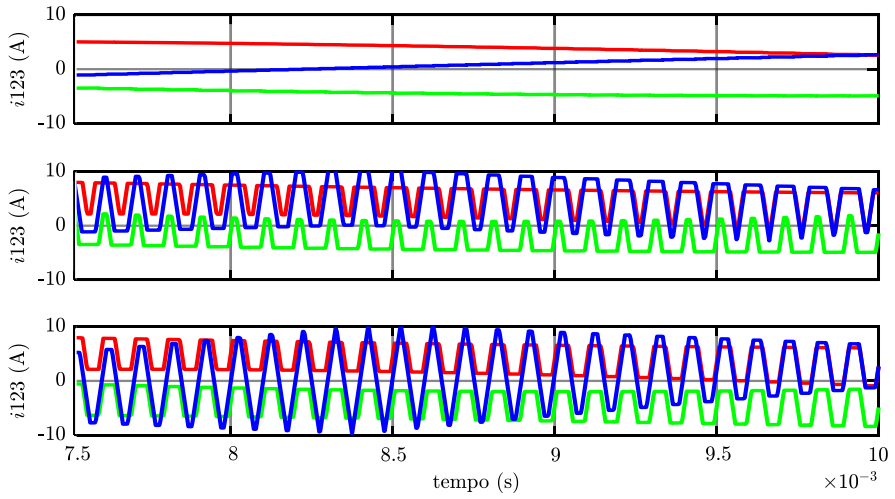


Figura 80 - Effetto dello sfasamento delle portanti sul ripple di i123 (0, 25, 50%)

3.2.6 Modulazione tipo BEM o Sinusoidale

Nella teoria deli controlli una tecnica di modulazione molto apprezzata è la tipo BEM che, introducendo delle terze armoniche sui duty cycle, permette di aumentarne il valore della fondamentale, consentendo così un migliore sfruttamento della tensione di DC-link. Si intende ora confrontare i risultati ottenibili con una modulazione di tipo sinusoidale rispetto a quelli conseguenti ad una modulazione tipo BEM; tali risultati si presentano in Figura 81Figura 82 e in Figura 82.

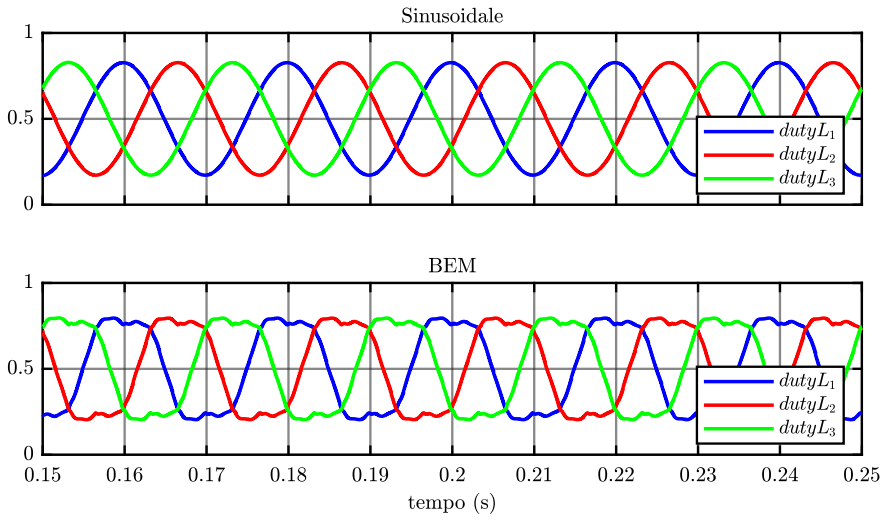


Figura 81 - Confronto modulazione BEM e Sinusoidale: duty cycle.

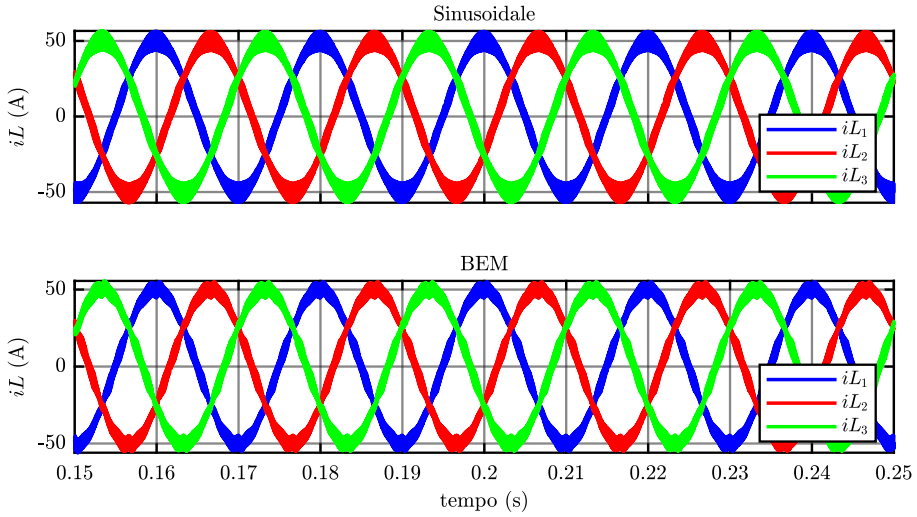


Figura 82 - Confronto modulazione BEM e Sinusoidale: corrente controllata.

Con riferimento alla Figura 82 si nota che la modulazione di tipo sinusoidale restituisce una migliore qualità delle forme d'onda in uscita. Con la modulazione BEM, generalmente preferibile poiché consente di sfruttare meglio la tensione del DC-link, si inietta infatti nel sistema della componente di modo comune che altrimenti non comparirebbe. Non avendo vincoli sulla tensione di DC-link si predilige pertanto la modulazione sinusoidale.

3.2.7 Filtro LCL

Si inserisce nel sistema, fra i due convertitori, un filtro LCL al fine di consentire la regolazione del ripple.

Per il dimensionamento dei componenti si fa riferimento ai valori massimi di tensione e corrente a cui questi verranno sottoposti in fase di collaudo. Per la capacità si considera che la massima tensione che si intende testare è $V_{DC,max} = 750\ V$, quindi la massima tensione efficace sul condensatore è:

$$V_{DC,rms} = \frac{750Vrms}{\sqrt{3} \cdot \sqrt{2}} = 300Vrms$$

La tensione di dimensionamento risulta $230Vrms$, ovvero il valore standard più vicino al risultato ottenuto.

Per il valore di induttanza si fa riferimento alla massima corrente che si intende provare. Tenendo conto dei componenti disponibili in magazzino in quanto a taglia a portata in corrente, si pone $L_f = 300uH$ e si verifica che una corrente di $110Arms$, cioè quella di interesse, sia raggiungibile. In definitiva si inserisce un banco di condensatori da 40 μF, con le relative resistenze serie di smorzamento, e due induttori da 300 μH.

Una volta inserito il filtro LCL può risultare necessario modificare il controllo dell'anello di corrente prendendo come riferimento la tensione sul condensatore e non più quella in ingresso al convertitore. È importante notare, infatti, che in un controllo vettoriale, come è quello in assi dqo, sorge la necessità di orientare gli assi di riferimento nella giusta direzione. La presenza del filtro fa sì che fra la corrente i_L e la tensione v_C nasca infatti un angolo di sfasamento a causa degli elementi reattivi introdotti.

3.2.8 Phase Locked Loop e SOGI

In seguito all'inserzione del filtro LCL non è più possibile usare come riferimento per il controllo l'angolo $theta\_REF$ generato internamente, ma si rende necessario ricavare la fase della nuova tensione di controllo, ovvero la tensione sui condensatori del filtro. Si implementa dunque un Phase Locked Loop, ovvero un anello inseguitore di fase che permette di ricavare istante per istante la fase di una data grandezza. Normalizzando rispetto al valore di picco le componenti della tensione in assi $\alpha\beta$ si ottengono il seno ed il coseno dell'angolo di interesse, permettendo si ricavarne l'ampiezza.

Note le componenti della tensione sul condensatore in assi $\alpha\beta$ si calcola dunque:

$$\frac{V_{C\alpha}}{\hat{V}_C} = cos\ theta\_PLL$$

$$\frac{V_{C\beta}}{\hat{V}_C} = sin\ theta\_PLL$$

Dove:

$$\hat{V}_C = \sqrt{{V_{C\alpha}}^2 + {V_{C\beta}}^2}$$

Infine, dal rapporto delle due quantità si ricava l'angolo come segue:

$$theta\_PLL = \text{atan}\left(\frac{sin\ theta\_PLL}{cos\ theta\_PLL}\right)$$

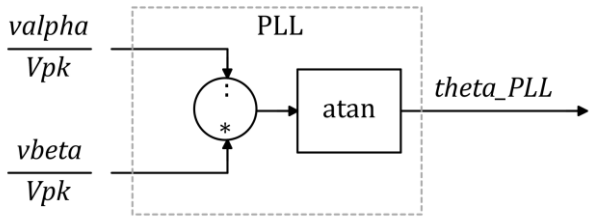


Figura 83 - Schema a blocchi di un PLL.

In Figura 84 si mostra la differenza di fase fra la tensione precedentemente utilizzata per la regolazione, ovvero la tensione v_L di fase $Theta\_REF$, e la tensione sul banco di condensatori di fase $Theta\_PLL$.

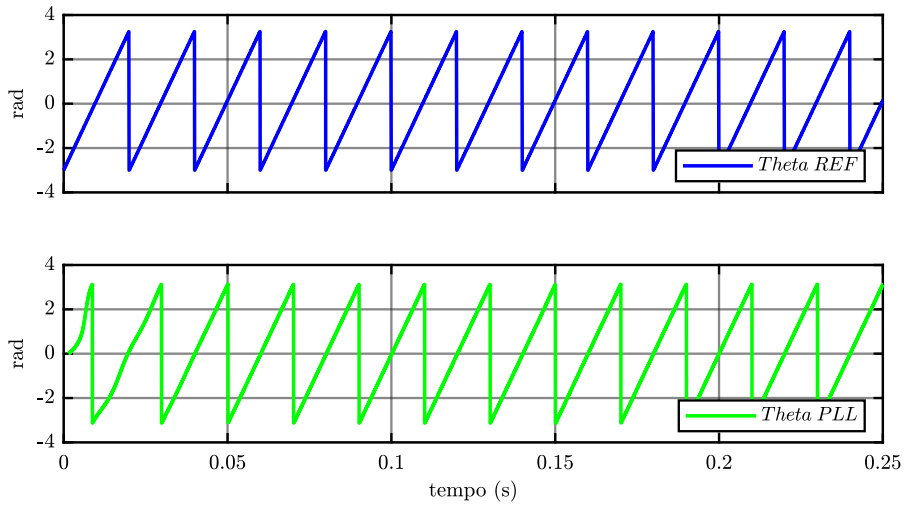


Figura 84 - Confronto fra Theta_REF e Theta_PLL.

A monte del PLL si sceglie inoltre di inserire un filtro risonante Second Order Generalized Integrator (SOGI), al fine di selezionare la sola armonica di interesse, ovvero la prima che si trova alla frequenza di riferimento, ed attenuare le restanti.

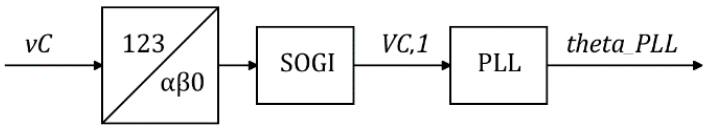


Figura 85 - Schema a blocchi di SOGI e PLL.

In Figura 86 si mostra lo schema a blocchi del controllo completo comprendente il blocco PLL e il filtro SOGI.

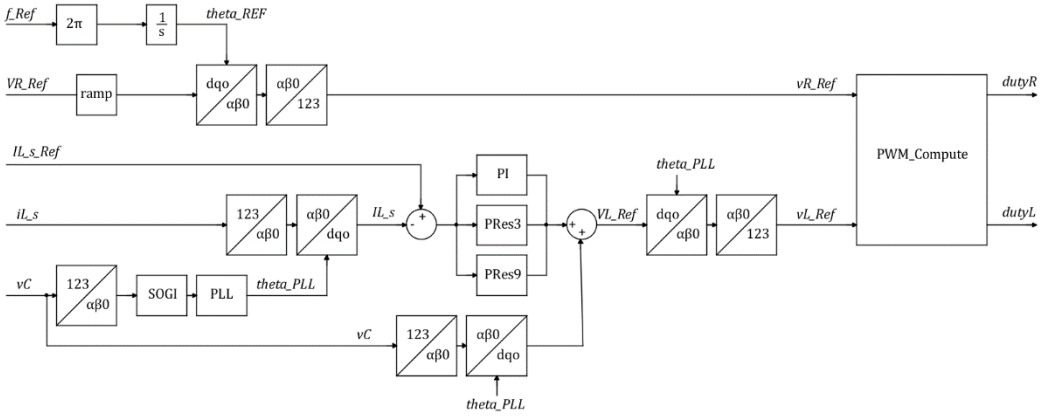


Figura 86 - Schema di controllo completo, con PLL.

Si verificano quindi gli effetti della presenza di PLL e SOGI sulla corrente controllata.

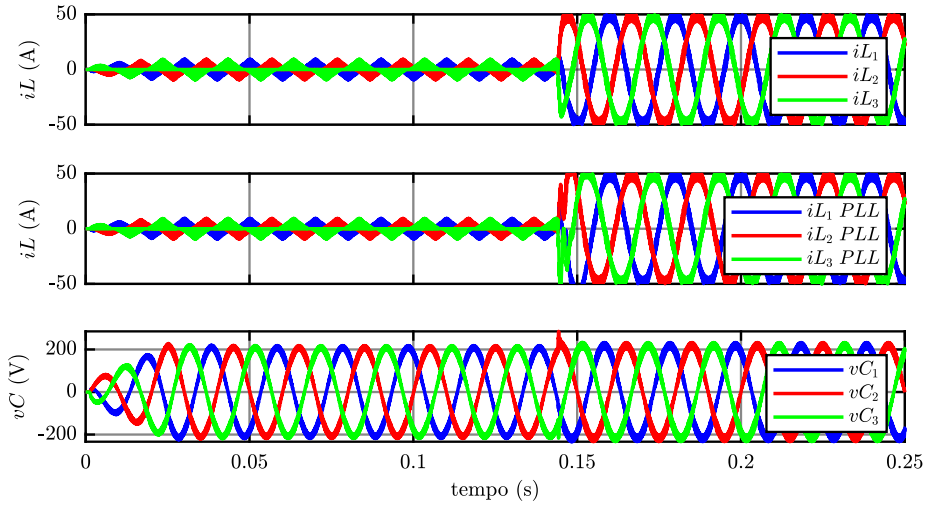


Figura 87 - Confronto tra controllo di corrente con e senza PLL.

In Figura 87 si osserva una perturbazione sulla corrente risultante dal controllo con PLL. Quando al tempo $t = 0{,}14\ s$ si chiude l'anello di corrente abilitando la modulazione dell'unità L, infatti, la tensione sui condensatori subisce anch'essa una perturbazione, causando una leggera variazione dell'angolo $Theta\_PLL$. Lo stesso non avviene nel caso in cui si utilizza come riferimento l'angolo $Theta\_REF$, poiché questo è generato per integrazione e quindi è continuo.

Poiché la presenza dei nuovi blocchi introdotto non apporta al sistema alcuna miglioria si decide di non includere gli stessi nel progetto finale.

3.2.9 Indipendenza delle modulazioni

In seguito all'inserimento del filtro LCL nasce la necessità di abilitare le modulazioni delle due unità in modo separato. Il corretto funzionamento del sistema richiede infatti l'avvenuta sincronizzazione prima che si possa avviare la modulazione dell'anello di corrente, mentre quella dell'anello di tensione va abilitata prima, in modo da consentire la sincronizzazione stessa.

Si osservi ora, infatti, in Figura 67 la presenza di due diversi segnali di abilitazione in ingresso al blocco *Inverter Modulator*. Mentre il segnale di abilitazione dell'unità R per il controllo di tensione viene inviato alla partenza, quello della modulazione dell'anello di corrente si attiva solo dopo le operazioni preliminari del sistema.

3.2.10 Macchina a stati

Alla luce di quanto esposto fino ad ora si implementa infine una sezione di gestione degli stati, che permetta di regolare l'esecuzione del codice scritto, per assicurare che gli eventi si susseguano nel giusto ordine ed in sicurezza.

Lo stato iniziale è, come sempre, quello di ERROR, nel quale tutte le variabili sono inizializzate, eventualmente azzerate, ed al quale si torna in seguito ad un qualunque evento di guasto.

A partire dallo stato di ERROR, si passa a quello di PRECHARGE quando si riceve il segnale esterno di $USER\_BTN$, impostato su dSPACE dall'utente. In questo stato si avvia la rampa della tensione di riferimento dell'anello di regolazione della tensione dell'unità R e se ne abilita la modulazione, portando ad 1 il segnale di $PWM\ EN\_R$.

Quando la rampa raggiunge il suo valore massimo si avviano il SOGI ed il PLL e se ne attende la sincronizzazione; ciò avviene nello stato di SYNC. Quando il PLL risulta essersi agganciato alla frequenza della tensione sul condensatore del filtro LCL, allora il sistema passa allo stato di READY, dal quale attende nuovamente il segnale $USER\_BTN$ per passare allo stato finale di START, nel quale si abilita la modulazione dell'anello di corrente.

In Figura 88 vengono rappresentati gli stati descritti e gli eventi chiave ad essi collegati.

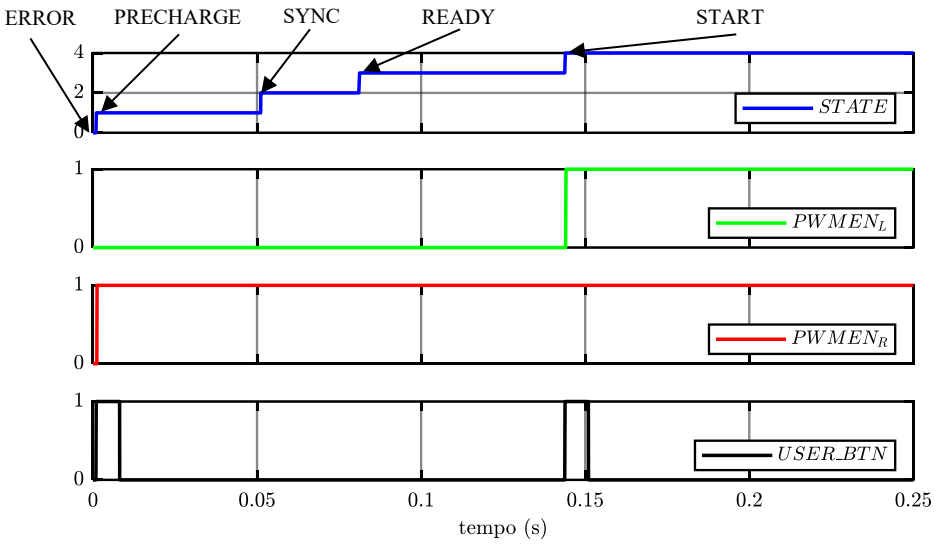


Figura 88 - Avanzamento della macchina a stati, segnali di abilitazione PWM e USER_BTN.

Capitolo 4

Implementazione su dSPACE

MicroLabBox di dSPACE è un sistema di sviluppo per il laboratorio che combina dimensioni compatte con alte prestazioni e versatilità. MicroLabBox consente di configurare le applicazioni di controllo, test e misura in modo rapido e semplice. Si tratta, infatti, di un sistema progettato e utilizzato principalmente per lo sviluppo di controllori digitali multi-variabili ad alta velocità e per simulazioni real-time in svariati campi di controllo tra cui gli azionamenti elettrici.

Figura 89 - dSPACE MicroLabBox [5].

Si utilizza quindi l'ambiente dSPACE per l'implementazione del codice di controllo ad alto livello degli inverter, precedentemente realizzato e testato sul software di simulazione PLECS. I codici di controllo per dSPACE dovranno inoltre includere una sezione per la comunicazione con l'FPGA, responsabile delle acquisizioni analogiche e della generazione dei segnali di gamba per mezzo del modulo PWM implementato. Le misure di tensione e corrente rappresentano infatti gli ingressi necessari al calcolo dei duty cycle, che avviene per mezzo del codice C scritto su PLECS e caricato su dSPACE, e i duty cycle, in uscita da questo, devono invece essere mandati all'FPGA che, tramite il modulo

PWM_Modulator implementato, realizza i segnali di controllo degli switch degli inverter.

I sistemi dSPACE dispongono infine di un'interfaccia real time (RTI – Real Time Interface) affiancabile all'applicazione Simulink di Matlab, per cui è possibile generare il codice real-time da utilizzare nel sistema dSPACE direttamente da un modello implementato su Simulink e caricarlo automaticamente nell'hardware. Gli opportuni codici di controllo vengono dunque scritti su Matlab e richiamati da un blocco S-Function di Simulink, il quale permette il collegamento diretto alla piattaforma dSPACE.

Di seguito si discutono dunque i codici di implementazione della comunicazione fra i due sistemi, quelli relativi al controllo testato su PLECS e la creazione dell'interfaccia a PC per l'utente.

4.1 Comunicazione con FPGA

Su Matlab si scrive il file.c per la gestione della comunicazione con FPGA. Questo risulta diviso in alcune sezioni principali:

- Inizializzazione porte e variabili;
- Lettura dati in ingresso;
- Elaborazione ingressi e calcolo delle uscite;
- Scrittura dati in uscita.

La prima parte del codice è dunque dedicata all'inizializzazione, nella quale vengono dichiarate le *word* in ingresso ed in uscita da dSPACE con le opportune impostazioni. A ciascun bit dei segnali inviati e ricevuti viene dunque assegnato un pin del connettore di collegamento fra l'FPGA e dSPACE, in accordo con la Tabella 2.

Di seguito si presentano, a titolo di esempio, le stringhe di codice per l'assegnazione dei pin di una parola a 6 Bit in uscita da dSPACE.

```
/* Create the driver object that is using channels 1 to 6 of port 1
of the DIO Class 1 unit. */
IErr=DioCl1DigOut_create(&WORD_OUT,DIO_CLASS1_PORT_1,DIO_CLASS1_MASK_
CH_1|DIO_CLASS1_MASK_CH_2|DIO_CLASS1_MASK_CH_3|DIO_CLASS1_MASK_CH_4|D
IO_CLASS1_MASK_CH_5|DIO_CLASS1_MASK_CH_6);
/* Initialize the DIO Class 1 unit. */
IErr=DioCl1DigOut_setSignalVoltage(WORD_OUT,DIO_CLASS1_SIGNAL_5_0_V);
/* Apply the default settings to the driver object. */
IErr=DioCl1DigOut_apply(WORD_OUT);
/* Enable the digital input channels. */
IErr=DioCl1DigOut_start(WORD_OUT);
```

Nella prima riga si osserva come assegnare, ad esempio, i pin da 1 a 6 della porta 1, che fanno parte di quelli preposti alla scrittura delle informazioni in uscita. Nelle righe successive vengono assegnate le impostazioni, in particolare la tensione del pin, e abilitati i canali scelti.

Tabella 2 - Assegnazione pin dSPACE per la comunicazione.

Signal	FPGA Name	Pin Name
WORD_IN [0]	BIT_OUT_DSPACE_0	DIO1 ch1
WORD_IN [1]	BIT_OUT_DSPACE_1	DIO1 ch2
WORD_IN [2]	BIT_OUT_DSPACE_2	DIO1 ch3
WORD_IN [3]	BIT_OUT_DSPACE_3	DIO1 ch4
WORD_IN [4]	BIT_OUT_DSPACE_4	DIO1 ch5
WORD_IN [5]	BIT_OUT_DSPACE_5	DIO1 ch6
WORD_IN [6]	BIT_OUT_DSPACE_6	DIO1 ch7
WORD_IN [7]	BIT_OUT_DSPACE_7	DIO1 ch8
WORD_IN [8]	BIT_OUT_DSPACE_8	DIO1 ch9
WORD_IN [9]	BIT_OUT_DSPACE_9	DIO1 ch10
WORD_IN [10]	BIT_OUT_DSPACE_10	DIO1 ch11
WORD_IN [11]	BIT_OUT_DSPACE_11	DIO1 ch12
dSPACE_EOP	BIT_OUT_DSPACE_12	DIO1 ch13
RD_WR	BIT_OUT_DSPACE_13	DIO1 ch14
R_dSPACE	BIT_OUT_DSPACE_14	DIO1 ch15
IRQ_dSPACE	BIT_IN_DSPACE_0	DIO1 ch20
WORD_OUT [0]	BIT_IN_DSPACE_1	DIO1 ch21
WORD_OUT [1]	BIT_IN_DSPACE_2	DIO1 ch22
WORD_OUT [2]	BIT_IN_DSPACE_3	DIO1 ch23
WORD_OUT [3]	BIT_IN_DSPACE_4	DIO1 ch24
WORD_OUT [4]	BIT_IN_DSPACE_5	DIO1 ch25
WORD_OUT [5]	BIT_IN_DSPACE_6	DIO1 ch26
WORD_OUT [6]	BIT_IN_DSPACE_7	DIO1 ch27
WORD_OUT [7]	BIT_IN_DSPACE_8	DIO1 ch28
WORD_OUT [8]	BIT_IN_DSPACE_9	DIO1 ch29
WORD_OUT [9]	BIT_IN_DSPACE_10	DIO1 ch30
WORD_OUT [10]	BIT_IN_DSPACE_11	DIO1 ch31
WORD_OUT [11]	BIT_IN_DSPACE_12	DIO1 ch32
WORD_OUT [12]	BIT_IN_DSPACE_13	DIO1 ch33
WORD_OUT [13]	BIT_IN_DSPACE_14	DIO1 ch34

In Tabella 2 si evidenziano in grassetto quattro segnali di particolare importanza, poiché sono quelli sui quali si basa la comunicazione. Di questi il segnale IRQ_dSPACE è l'unico in ingresso e viene infatti generato dal modulo IRQ_Manager dell'FPGA e mandato a dSPACE come interrupt. Per quanto riguarda gli altri tre segnali, $dSPACE\_EOP$ è quello che permette l'inizio e la fine

della comunicazione, RD_WR indica se il dato in comunicazione è di lettura o di scrittura e infine R_dSPACE scandisce le operazioni effettuate su ogni parola. In Figura 90 si mostra la sequenza dei valori assunti dai quattro segnali per una corretta comunicazione.

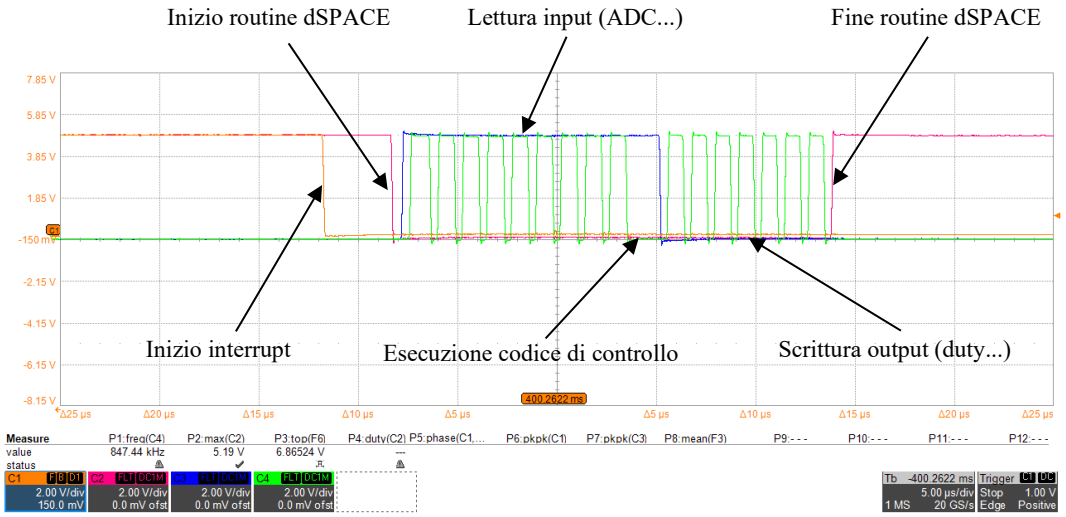


Figura 90 - Sequenza dei segnali per comunicazione.

Il segnale arancione in Figura è l'interrupt, all'interno del quale devono avvenire le operazioni di lettura, calcolo e scrittura. In rosso è indicato il segnale $dSPACE\_EOP$ che abilita la comunicazione se portato allo stato logico basso. Il segnale in blu è il RD_WR che consenta la lettura se alto e la scrittura se basso. Il segnale R_dSPACE , verde in Figura, abilita la trasmissione dei dati e viene infatti portato allo stato logico 1 ad ogni word in ingresso e in uscita; si osservano allora i nove impulsi corrispondenti alle nove parole in lettura e i sette corrispondenti alle parole in scrittura, in accordo con la schematizzazione di Figura 51.

Di seguito si mostrano le stringhe di codice, eseguite ad ogni interrupt, che portano il $dSPACE\_EOP$ al livello basso, al fine di dare inizio la comunicazione, e il RD_WR al livello alto per abilitare la prima operazione, cioè quella di lettura.

```
#ifndef MATLAB_MEX_FILE
    /* Write the output states to the internal buffer. */
    IErr=DioCl1DigOut_setChMaskOutData(dSPACE_EOP,0x00);
    /* Output the data at the output channels. */
    IErr=DioCl1DigOut_write(dSPACE_EOP);

    /* Write the output states to the internal buffer. */
    IErr=DioCl1DigOut_setChMaskOutData(RD_WR,0x80);
    /* Output the data at the output channels. */
    IErr=DioCl1DigOut_write(RD_WR);
#endif
```

Si procede allora alla lettura dei dati tramite la funzione "*ReadingWORD_IN*", come segue:

```
//Reading Part
ReadingWORD_IN(&data_INA);
```

Una volta applicata la precedente funzione a tutti i dati trasmessi a dSPACE, si decodificano opportunamente tali dati come mostrato di seguito.

```
//Translating information from data IN

Iabc.a= (((double)data_INA) - IabcOfs.a)* IabcK.a;
Iabc.b= (((double)data_INB) - IabcOfs.b)* IabcK.b;
Iabc.c= (((double)data_INC) - IabcOfs.c)* IabcK.c;

Iabc_R.a= (((double)data_IND) - IabcOfs.a)* IabcK.a;
Iabc_R.b= (((double)data_INE) - IabcOfs.b)* IabcK.b;
Iabc_R.c= (((double)data_INF) - IabcOfs.c)* IabcK.c;

Vdc_L = (((double)data_ING) - VdcLOfs)* VdcLK;
Vdc_R = (((double)data_INH) - VdcROfs)* VdcRK;

overcurrent1A = (double)(data_INI&0x01);
overcurrent1B = (double)(data_INI&0x02);
overcurrent1C = (double)(data_INI&0x04);
overcurrent2A = (double)(data_INI&0x08);
overcurrent2B = (double)(data_INI&0x10);
overcurrent2C = (double)(data_INI&0x20);
overvoltage_DC1 = (double)(data_INI&0x40);
overvoltage_DC2 = (double)(data_INI&0x80);
```

Si osserva che ai dati relativi alle acquisizioni, contenuti nelle WORD_IN da A ad H, viene applicato un offset per eliminare appunto eventuali errori di offset del circuito di misura analogica, ed un opportuno fattore di scala che tiene conto del circuito di condizionamento della relativa catena di acquisizione. Tali offset vengono calcolati in uno stato aggiuntivo rispetto a quelli presentati al paragrafo 3.2.10, denominato WAKE_UP e posto fra gli stati ERROR e PRECHARGE.

L'ultimo dato in ingresso contiene l'informazione degli errori rilevati dal modulo di protezione hardware dell'FPGA. Poiché ad ogni bit corrisponde un tipo di errore, si applicano delle maschere in codice esadecimale che permettono di isolare il bit di interesse ed assegnare alle variabili *overcurrentxy* e *overvoltage_DCx* il valore 1 o 0 a seconda che il guasto sia presente o meno.

Una volta che tutte le misure necessarie al controllo sono state acquisite si procede con l'esecuzione della funzione *SemikronControl* che contiene il codice implementato per PLECS nel blocco C-Script.

```
SemikronControl();
```

Tale funzione restituisce in uscita i valori dei duty cycle di ogni gamba di ciascun inverter in formato double. A questi segnali si applica dunque un'ulteriore funzione chiamata *DutyConv* al fine di adattarli alle necessità del codice di

comunicazione e del modulatore FPGA. Di seguito si mostra l'implementazione del condizionamento del duty cycle, ad esempio, della fase a.

```
tmpDutyConv = dutyDouble->a - 0.5;
    tmpDutyConv = tmpDutyConv*(2047.0);
    if (dutyDouble->a>0.5){
        dutyInt = ceil(tmpDutyConv);
        *dutyA = dutyInt;
        }
    else {
        dutyInt = floor(tmpDutyConv);
        dutyInt32 = dutyInt & (0x00000FFF);
        *dutyA = dutyInt32;
        }
```

Il segnale calcolato in *SemikronControl* è infatti un dato in formato double che varia fra 0 e 1, ma il modulatore dell'FPGA è pensato per operare con ingressi compresi fra $-0{,}5$ e $+0{,}5$, in quanto possiede un generatore di triangola bipolare. I duty cycle vengono allora shiftati di 0,5 e riscalati per adattarli al numero di bit con cui lavora il blocco modulatore, trasformati in numeri interi per conformarli ai requisiti della funzione di scrittura della comunicazione e, infine, viene loro applicata una maschera che serve ad isolare i soli 12 bit che devono essere inviati all'FPGA.

Si preparano dunque i dati da mandare in uscita da dSPACE, assegnando le variabili di interesse ai segnali *data_OUT* come segue:

```
//Translating information to data OUT
DutyConv(&dutyL, &dutyAL, &dutyBL, &dutyCL);
DutyConv(&dutyR, &dutyAR, &dutyBR, &dutyCR);
data_OUTA=dutyAL;
data_OUTB=dutyBL;
data_OUTC=dutyCL;
data_OUTD=dutyAR;
data_OUTE=dutyBR;
data_OUTF=dutyCR;

CONF_to_FPGA=0x0000;
CONF_to_FPGA |= ((UInt32)PWM_EN_L);
CONF_to_FPGA |= (((UInt32)PWM_EN_R)<<1);
CONF_to_FPGA |= (((UInt32)Fault_SW)<<2);
CONF_to_FPGA |= (((UInt32)Reset_HW)<<3);
CONF_to_FPGA |= (((UInt32)CMD_RL)<<4);

data_OUTG=CONF_to_FPGA;
```

Con logica del tutto analoga a quella utilizzata per la decodifica effettuata sul dato contenente il registro di errori hardware, si opera adesso una codifica del registro configurazioni da mandare all'FPGA. Le precedenti righe di codice servono infatti ad assegnare il valore delle variabili binarie alla destra delle uguaglianze ad un determinato bit del segnale $CONF\_to\_FPGA$, che viene dunque caricato sul *data_OUTG*. Quest'ultimo conterrà dunque le abilitazioni dei modulatori dei due inverter, il segnale di errore in ingresso al modulo di protezione degli switch e il segnale di reset da mandare al modulo di protezione hardware.

Infine, dopo aver portato il segnale RD_WR al livello basso, si procede alla scrittura dei dati tramite la funzione "*WritingWORD_OUT*".

```
#ifndef MATLAB_MEX_FILE
        /* Write the output states to the internal buffer. */
        IErr=DioCl1DigOut_setChMaskOutData(RD_WR,0x00);
        /* Output the data at the output channels. */
        IErr=DioCl1DigOut_write(RD_WR);
    #endif


//Writing part
WritingWORD_OUT(data_OUTA);
```

Alla fine del processo si riporta il $dSPACE\_EOP$ ad 1 e si lasciano il RD_WR il R_dSPACE a 0.

```
    #ifndef MATLAB_MEX_FILE
        /* Write the output states to the internal buffer. */
        IErr=DioCl1DigOut_setChMaskOutData(dSPACE_EOP,0x40);
        /* Output the data at the output channels. */
        IErr=DioCl1DigOut_write(dSPACE_EOP);

        /* Write the output states to the internal buffer. */
        IErr=DioCl1DigOut_setChMaskOutData(RD_WR,0x00);
        /* Output the data at the output channels. */
        IErr=DioCl1DigOut_write(RD_WR);

        /* Write the output states to the internal buffer. */
        IErr=DioCl1DigOut_setChMaskOutData(R_dSPACE,0x0);
        /* Output the data at the output channels. */
        IErr=DioCl1DigOut_write(R_dSPACE);
    #endif
```

4.2 Codice di controllo

Nel file.c viene inserita una Macchina a Stati che gestisce la sequenza delle operazioni del sistema. Questa riassume tutti gli stati utilizzati nel codice VHDL di Vivado e in quello C di PLECS, rendendo quindi superflue le precedenti due FSM implementate. Il nuovo codice fornisce infatti all'FPGA le abilitazioni, gli errori e i reset necessari tramite un apposito registro $CONF\_to\_FPGA$ ed escorpora le operazioni precedenti dallo stato START dal controllo sviluppato in PLECS, ora racchiuso nella funzione *SemikronControl*.

Si parte come da consuetudine dallo stato di ERROR, nel quale tutte le variabili e i parametri vengono inizializzati.

```
switch(STATE) {
  case ERROR:
  …
  …

if(USER_BUTTON > 0.5){
    if (flagOfs == 0)
```

```
            STATE = WAKE_UP;
        else
            STATE = PRECHARGE;
    }
    counterOfs = 0;
break;
```

Quando viene cliccato lo user button dall'utente, si passa allora allo stato di WAKE_UP, dove avviene il calcolo degli offset delle acquisizioni analogiche a cui si era accennato nel paragrafo precedente. Questa operazione viene fatta una sola volta all'accensione del sistema, infatti, alla fine dello stato un flag, denominato "flagOfs, viene attivato in modo da non ripetere l'operazione qualora il sistema dovesse tornare in ERROR. Di seguito si mostra il calcolo degli offset.

```
case WAKE_UP:
  IabcOfs.a += data_INA;
  IabcOfs.b += data_INB;
  IabcOfs.c += data_INC;
  VdcLOfs += data_IND;
  VdcROfs += data_INE;
  if(counterOfs == 1000)
  {
      IabcOfs.a = IabcOfs.a/counterOfs;
      IabcOfs.b = IabcOfs.b/counterOfs;
      IabcOfs.c = IabcOfs.c/counterOfs;
      VdcLOfs = VdcLOfs/counterOfs;
      VdcROfs = VdcROfs/counterOfs;
  }
  else
      counterOfs++;
  flagOfs = 1;
  CMD_RL = 0.0;
  STATE = PRECHARGE;
break;
```

Si sottolinea in questa occasione che le schede VAB presentano lungo la linea di acquisizione dei relè K1 IM01TS [1]. Questi vengono chiusi solo dopo l'operazione di calcolo degli offset, in modo da permettere un calcolo più preciso e non affetto dall'interferenza dell'alimentazione.

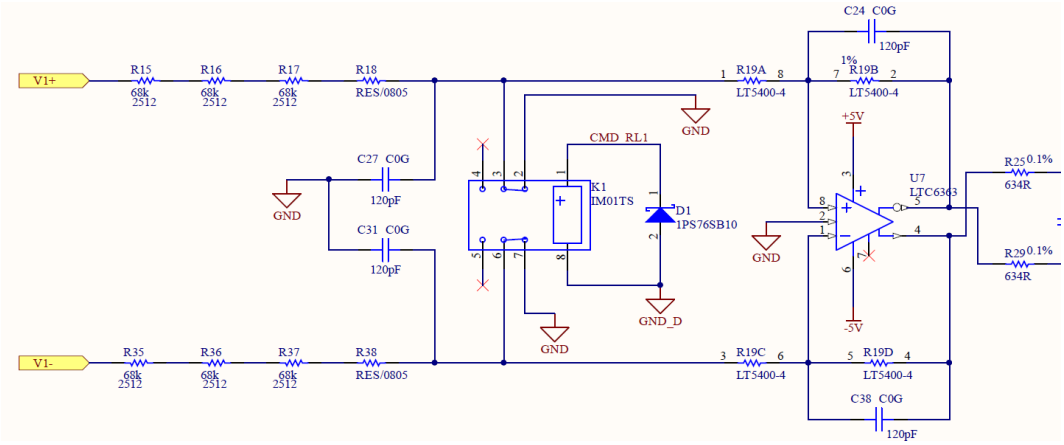


Figura 91 - K1 IM01TS.

Il relè in questione è di tipo solo SET, cioè alimentando opportunamente la bobina posta fra i morsetti 1 e 2 i contatti si spostano dai pin 3-2 e 6-7 ai pin 3-4 e

6-5, in modo da rimuovere i collegamenti a ground e lasciare che il segnale in ingresso passi attraverso la linea di acquisizione. Come mostrato nelle righe di codice precedenti, l'alimentazione della bobina si ottiene grazie alla definizione di un flag CMD_RL il cui stato viene mandato all'FPGA e successivamente trasmesso alle schede di acquisizione di tensione, tramite l'apposito connettore.

Successivamente si percorrono gli stati di PRECHARGE, SYNC e READY, esposti nel paragrafo dedicato del precedente capitolo.

```
 case PRECHARGE:
counter++;

V_Slope  = 5000.0*Ts;
SlewRateLimiter( &VdR_Target , &V_Slope , &Vdq_R_ref.d);
PWM_EN_L = 0.0;
PWM_EN_R = 1.0;

if (counter>500){
counter  = 0;
STATE  = SYNC;
}
break;

case SYNC:
counter++;
if (counter>300){
 counter = 0;
 STATE = READY;
}
break;

case READY:
if ( USER_BUTTON>0.5)
  STATE = START;
break;
```

Infine con un secondo click sullo user button si giunge allo stato START, nel quale viene chiamata la routine di controllo degli anelli di regolazione dei due inverter.

```
 case START:
    PWM_EN_L = 1.0;
   SemikronControl();
   break;
    }
```

4.3 Interfaccia dSPACE – Control Desk

Il software ControlDesk della dSPACE è l'interfaccia grafica a PC tra il sistema e l'utente che offre un unico ambiente di lavoro, dall'inizio della sperimentazione fino alla fine. Questo permette di memorizzare l'andamento delle variabili presenti nell'applicazione real-time, di visualizzare su schermo l'andamento delle grandezze desiderate mediante un apposito strumento e di modificare in tempo reale i parametri dell'applicazione.

Come anticipato, l'interfaccia real time interagisce con Matlab tramite Simulink grazie alla libreria di blocchi di I/O messi a disposizione dalla dSPACE per configurare l'hardware della scheda. In Figura 92 si mostra lo schema Simulink utilizzato.

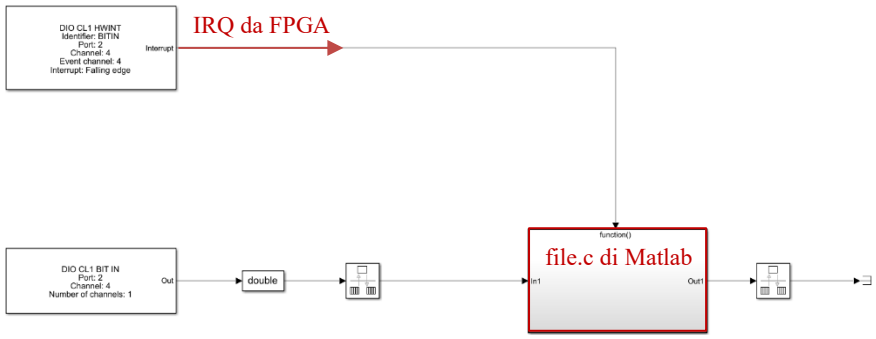


Figura 92 - Modello Simulink per dSPACE.

Tramite un apposito file.trc è possibile importare sul ControlDesk le variabili ed i parametri da visualizzare e modificare. Su tale file si indicano infatti all'interno di opportuni gruppi, ad esempio "*CONF_REG*", i segnali del codice che si ha interesse ha interesse a visualizzare come "READONLY" e quelli che si intende modificare come "PARAM". Per ciascuno si indica poi se la variabile è intera o di tipo float, rispettivamente con le diciture int(32) o flt(64,IEEE). Di seguito si presenta un esempio della definizione di un gruppo all'interno della *user trace*.

```
group "CONF_REG"
USER_BUTTON{
 type: flt(64,IEEE)
 alias: "USER_BUTTON"
 flags: PARAM
}
STATE{
 type: int(32)
 alias: "STATE"
 flags: PARAM
}
PWM_EN_L{
 type: flt(64,IEEE)
 alias: "PWM_EN_L"
 flags: PARAM
}
PWM_EN_R{
 type: flt(64,IEEE)
 alias: "PWM_EN_R"
 flags: PARAM
}
Fault_SW{
 type: flt(64,IEEE)
 alias: "Fault_SW"
 flags: PARAM
}
Reset_HW{
 type: flt(64,IEEE)
 alias: "Reset_HW"
 flags: PARAM
}
endgroup
```

ControlDesk è suddiviso in varie sezioni, fra cui la *Work Area*, che è l'area grafica a disposizione per realizzare interfacce grafiche, denominate layout, con gli strumenti desiderati. Si configurano allora i layout necessari all'utilizzo, all'interno dei quali inserire i segnali dichiarati nel file.trc, che vengono di seguito esposti.

Nella schermata principale, chiamata "Main" e rappresentata in Figura 93, si inserisce uno strumento Led che permette di visualizzare lo stato corrente della macchina a stati scritta nel file.c e che cambia colore in accordo allo stato indicato alla destra del led. Si inseriscono quindi due strumenti Button, corrispondenti ai segnali di USER_BUTTON, per l'avanzamento di stato, e di ERROR, per il ritorno manuale allo stato omonimo. Due Led vengono predisposti per la visualizzazione delle abilitazioni delle modulazioni dei due inverter e altri otto led vengono inseriti per indicare la presenza di uno dei possibili errori hardware rilevabili.

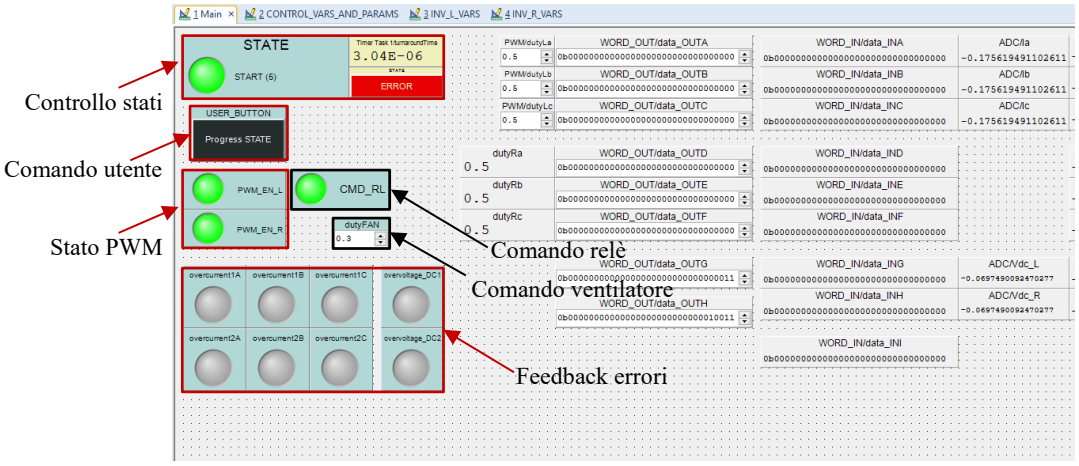


Figura 93 - Main di ControlDesk.

Il successivo layout, mostrato in Figura 94, permette di visualizzare le grandezze controllate in assi abc e dqo ed i loro riferimenti in assi dqo, tramite lo strumento Time Plotter. Vengono poi inseriti dei Numeric Input, in azzurro, e Display, in grigio, rispettivamente per i parametri da controllare e le variabili da visualizzare.

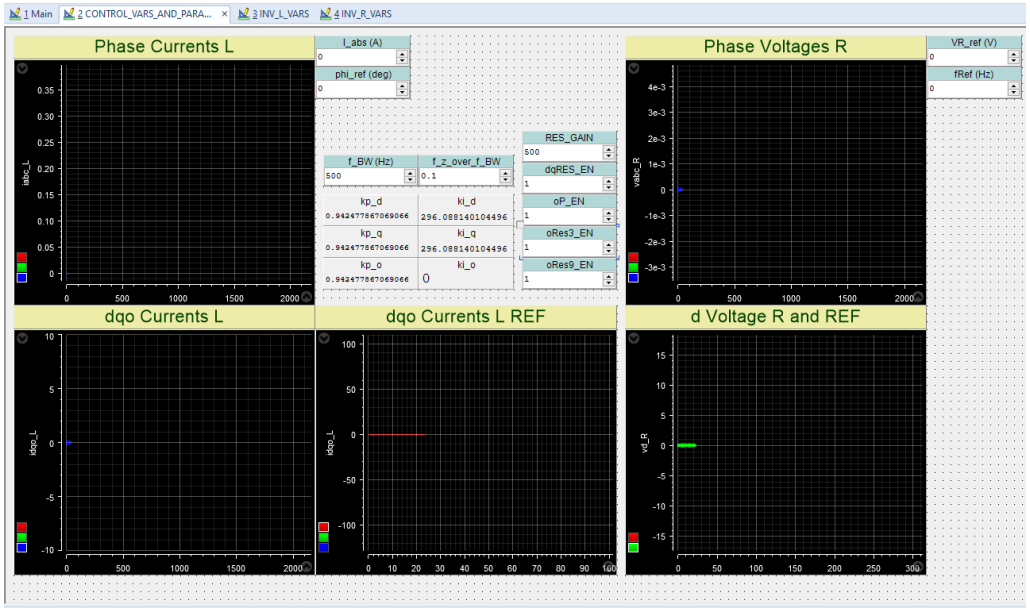


Figura 94 - Variabili e parametri del controllo di ControlDesk.

In Figura 95 e in Figura 96 si presentano i layout atti alla visualizzazione delle correnti AC e tensioni DC acquisite da ciascun inverter ed i rispettivi duty cycle calcolati per le tre fasi.

Figura 95 - Variabili dell'unità L di ControlDesk.

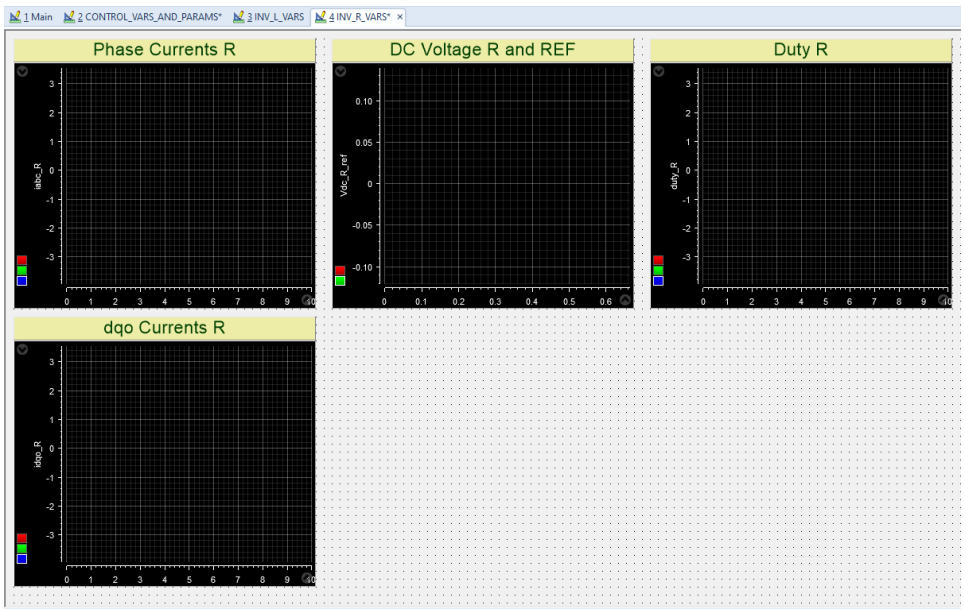


Figura 96 - Variabili dell'unità R di ControlDesk.

Capitolo 5

Assemblaggio meccanico

In questo capitolo vengono descritte le fasi dell'assemblaggio meccanico dell'intero sistema. Questo risulta composto, oltre che dai due stack Semikron, da una scheda FPGA, per la gestione del codice di basso livello, dalla piattaforma dSPACE, sulla quale viene caricato il codice di alto livello e dalle schede di misura e di interfaccia inverter.

In fase di assemblaggio si procede allora alla saldatura dei componenti sulle PCB delle schede e alla realizzazione degli appositi connettori per collegare le varie parti del sistema; si realizzano infine i disegni quotati di alcuni supporti in alluminio necessari per il l'assemblaggio di tutte le parti

5.1 Assemblaggio schede

Si ricorda che per rendere il dispositivo operativo è necessario disporre di schede di misura e di controllo appositamente progettate per l'interfaccia con i due inverter SEMIKRON [11].

Con riferimento alla Figura 8 si ricorda inoltre che il controllo è formato da una scheda principale FPGA da cui partono delle ramificazioni simmetriche che portano alle schede di misura di tensione e corrente e ad un'interfaccia inverter; vi sono poi la scheda di interfaccia relè e la scheda di interfaccia encoder, che non verranno però utilizzate nell'ambito di questa tesi.

Delle schede sono forniti: il circuito stampato (PCB – Printed Circuit Board), di cui si mostra un esempio in Figura 101, i componenti e la distinta dei materiali (BOM - Bill of Materials), in accordo alla quale saldare i componenti sulle schede.

Il processo di assemblaggio consta di alcuni passi principali:

- Stesura della pasta saldante;
- Posizionamento dei componenti SMD;
- Cottura in forno di reflow;
- Saldatura manuale dei componenti TTH, SMD sottostanti e dei connettori.

La pasta saldante deve essere stesa in corrispondenza delle piste dei componenti SMD (Surface Mounted Devices). A questo scopo si utilizza una maschera, ovvero una lamina che presenta dei fori in corrispondenza dei punti nei quali applicare la pasta. Tale lamina viene messa in tensione tramite un apposito telaio che ha l'ulteriore funzione di permettere che la maschera si allinei perfettamente al circuito stampato, come mostrato in Figura 97.

Figura 97 - Telaio con lamina e PCB.

A questo punto si possono posizionare i componenti sul circuito stampato, secondo quanto riportato sulla BOM.

Una volta terminato di disporre i componenti la scheda è pronta per essere infornata nel forno di reflusso. Tale forno è caratterizzato da una distribuzione del calore tramite convenzione forzata, secondo cicli di riscaldamento impostabili da PC e controllati mediante un microprocessore. Durante il ciclo di cottura la pasta saldante fonde e risolidifica permettendo il collegamento elettrico e meccanico dei

componenti alla PCB. Nella Figura 98 si mostra il ciclo utilizzato per la cottura, mentre in Figura 99 si presenta il forno di reflusso.

In Figura 102 si mostra infine il risultato della cottura sulla scheda FPGA, sulla quale si possono distinguere i componenti SMD saldati.

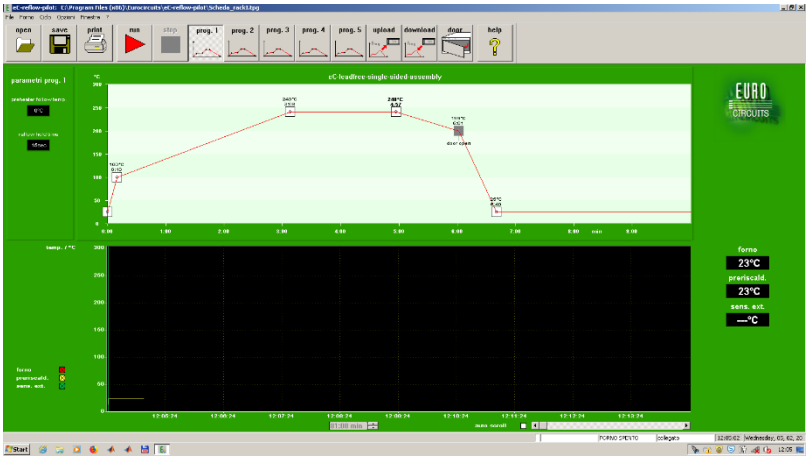


Figura 98 - Ciclo di cottura in forno di reflow.

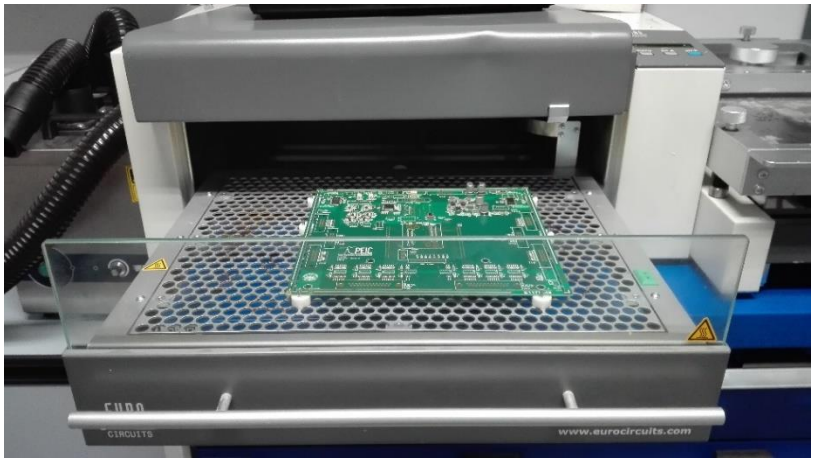


Figura 99 - Forno di reflow.

L'ultimo passo consiste infine nella saldatura manuale dei componenti a filo (TTH - through-hole technology) e SMD sottostanti la scheda. Come per il procedimento precedentemente illustrato lo scopo è nuovamente quello di realizzare la continuità elettrica e di fissare meccanicamente i pezzi alla PCB. A tale scopo si utilizza: una stazione saldante con controllo della temperatura impostato a 350°C, un rotolo di stagno per elettronica e una calza di rame per dissaldare in caso di errori.

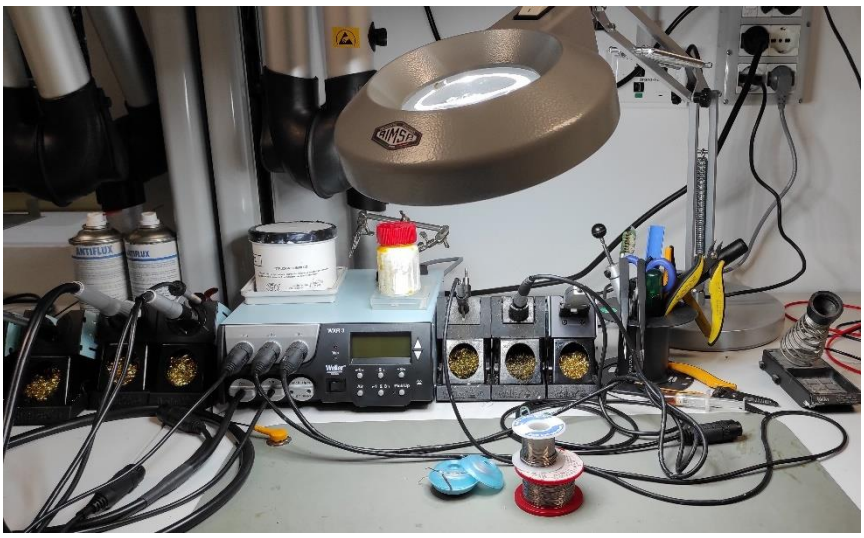


Figura 100 - Stazione saldante e attrezzatura per saldatura.

In Figura 103 si mostra la scheda FPGA finita, sulla quale sono stati saldati tutti i componenti elencati nella BOM.

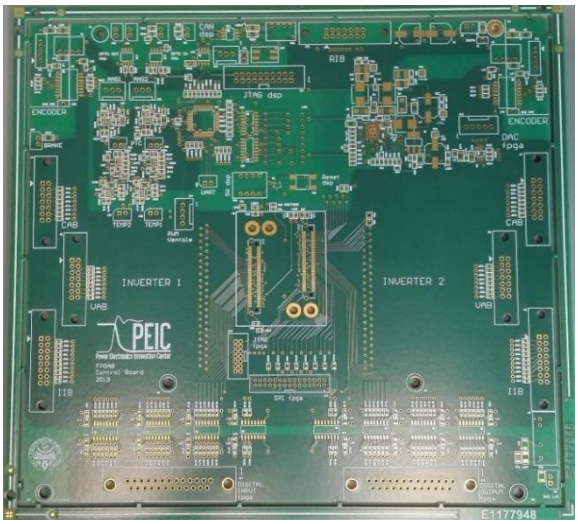


Figura 101 - PCB scheda FPGA.

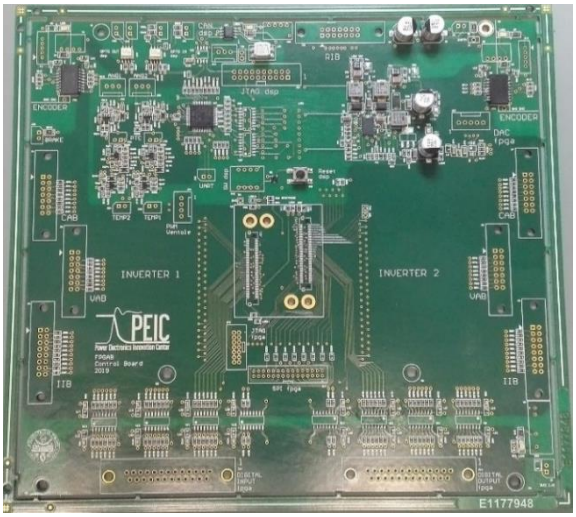


Figura 102 - Scheda FPGA con componenti SMD.

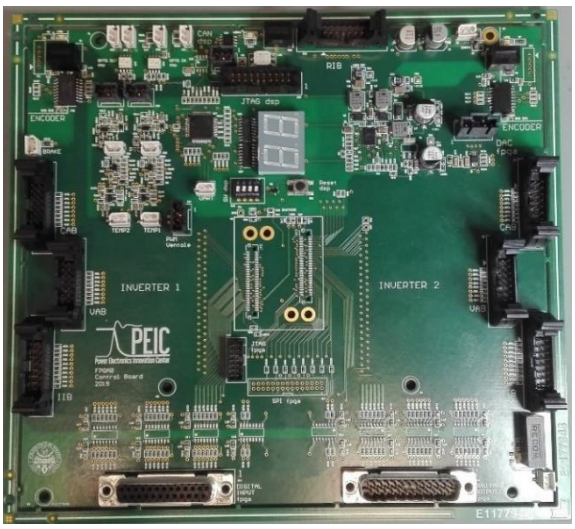


Figura 103 - Scheda FPGA completa.

Analogamente a quanto fatto per la scheda FPGA, si assemblano le schede di misura di corrente e tensione dei due stack, le schede di interfaccia inverter e interfaccia relè. Nella Figura seguente si mostra il risultato delle schede assemblate e disposte secondo lo schema dell'architettura mostrato in Figura 8.

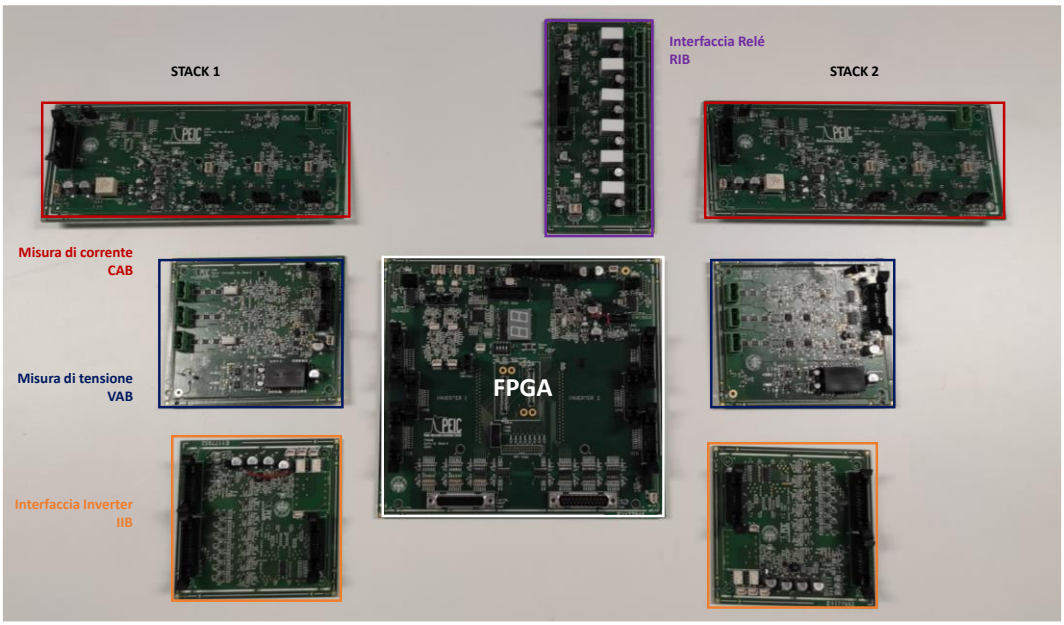


Figura 104 - Insieme delle schede assemblate.

5.2 Realizzazione connettori

Sulle schede assemblate, mostrate nel loro insieme in Figura 104, sono presenti alcuni connettori la cui funzione è quella di collegare le schede di interfaccia alla scheda FPGA e, nel caso delle schede IIB ad esempio, anche alla Driver Board dell'inverter, mostrata in Figura 3.

5.2.1 Connettori FPGA

Si fa riferimento agli schematici di ciascuna scheda e se ne studia il pinout per realizzare gli appositi connettori.

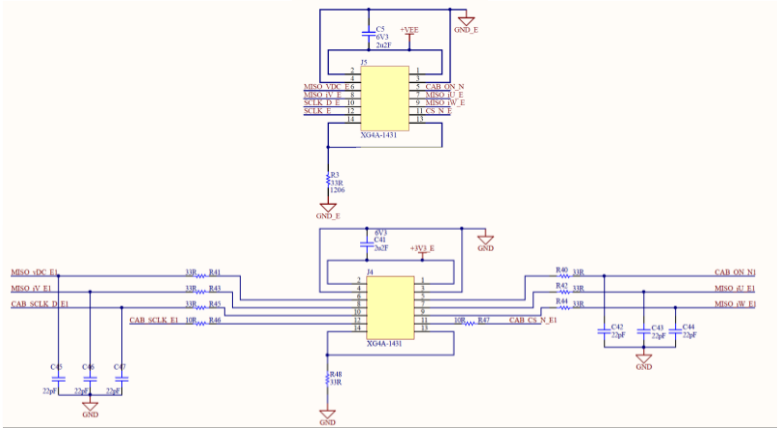


Figura 105 - Schema dei connettori: in alto scheda CAB, in basso scheda FPGA.

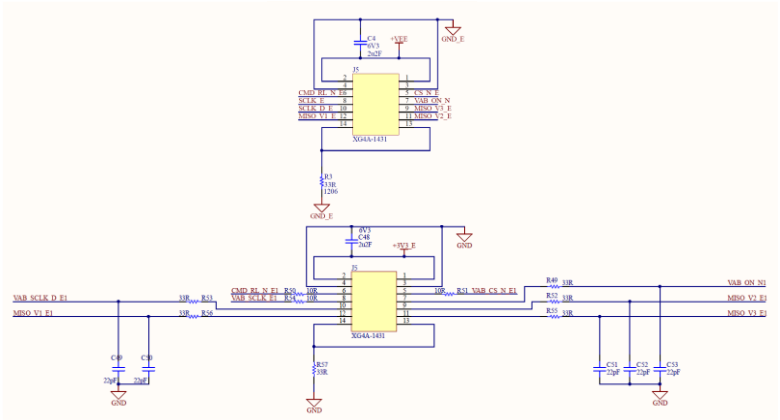


Figura 106 - Schema dei connettori: in alto scheda VAB, in basso scheda FPGA.

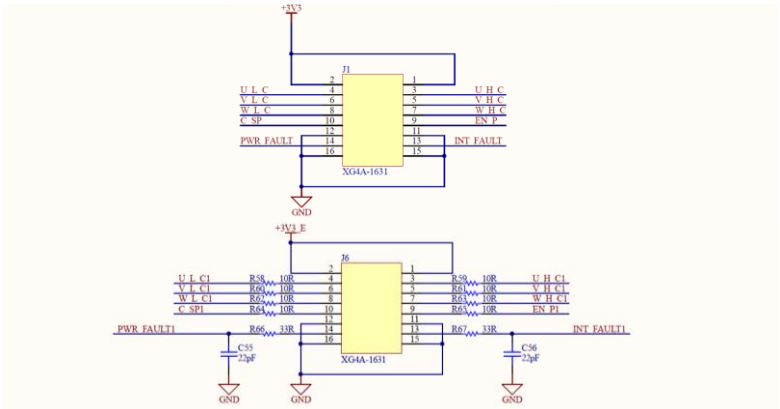


Figura 107 - Schema dei connettori: in alto scheda IIB, in basso scheda FPGA.

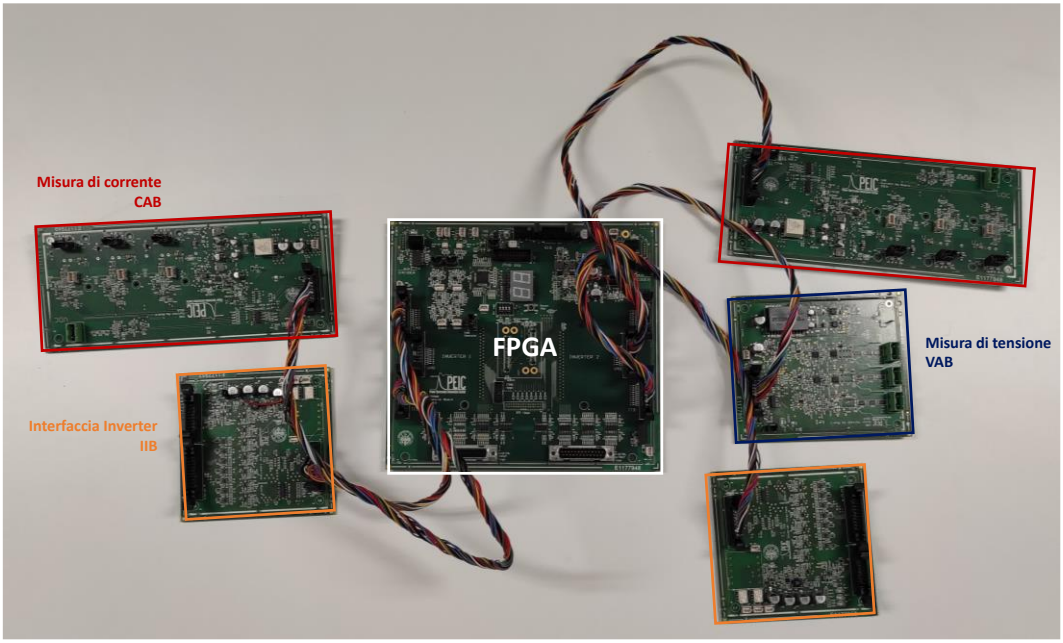


Figura 108 - Insieme delle schede utili assemblate e collegate fra loro.

5.2.2 Connettori Inverter

Si preparano poi i connettori per collegare la scheda IIB alla scheda driver degli inverter Semikron [12]. In Figura 109 si mostra lo schema dei segnali in ingresso ed in uscita dalla scheda driver in questione.

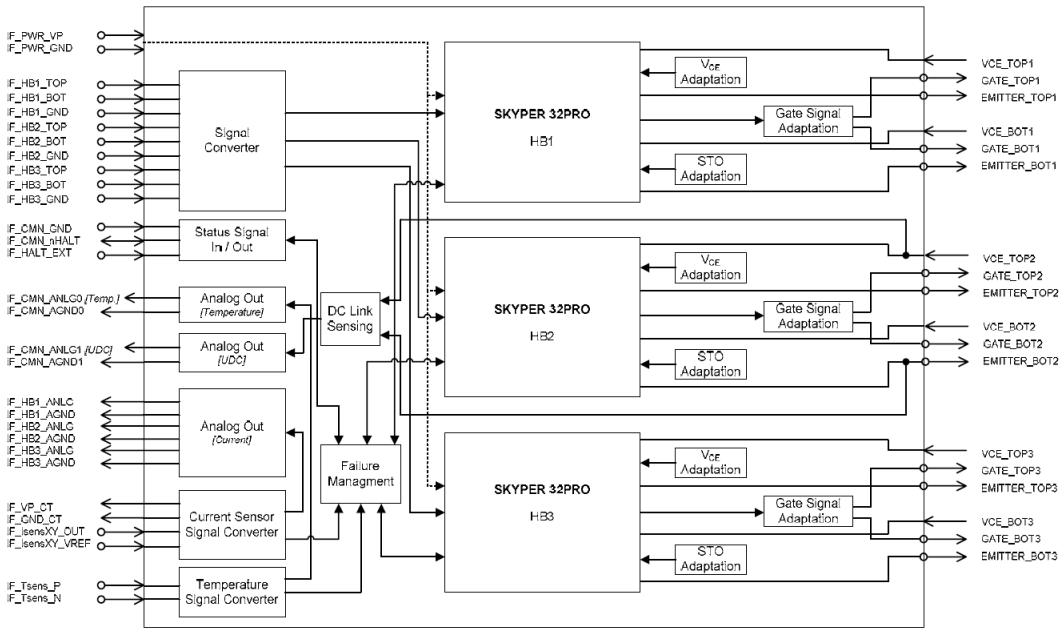


Figura 109 - SEMIKUBE Board GD 11 [12].

In Figura si osservano sulla sinistra i segnali richiesti dalla scheda per il suo corretto funzionamento, i quali vengono mandati dalla scheda IIB tramite

l'apposito connettore. Tra i più importanti si evidenziano i segnali di alimentazione, denominati IF_PWR_VP, e relativi ground IF_PWR_GND.

IF_CMN_nHALT è il segnale che, in logica negata, indica che la scheda è pronta ad operare, ma che viene messa in funzione solo quando il segnale IF_HALT_EXT è alto.

Vi sono poi i segnali IF_HBx_TOP e IF_HBx_BOT con i relativi ground, che sono i comandi di gamba, rispettivamente alto e basso, per le tre gambe. Si osservano infine i segnali IF_Tsens_P e IF_Tsens_N per le misure di temperatura, mentre i segnali delle misure analogiche di corrente IF_Isensx non vengono utilizzati poiché per queste vi è una scheda apposita.

Nelle successive figure si mostrano gli schemi dei connettori lato IIB, per mezzo dei quali avviene il collegamento con la scheda driver dell'inverter.

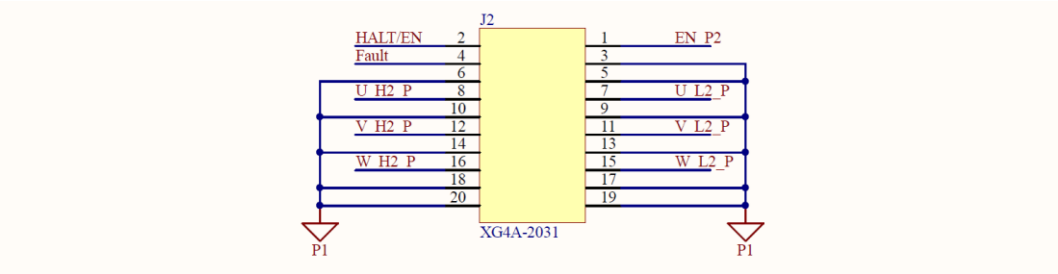


Figura 110 - Connettore J2 scheda IIB.

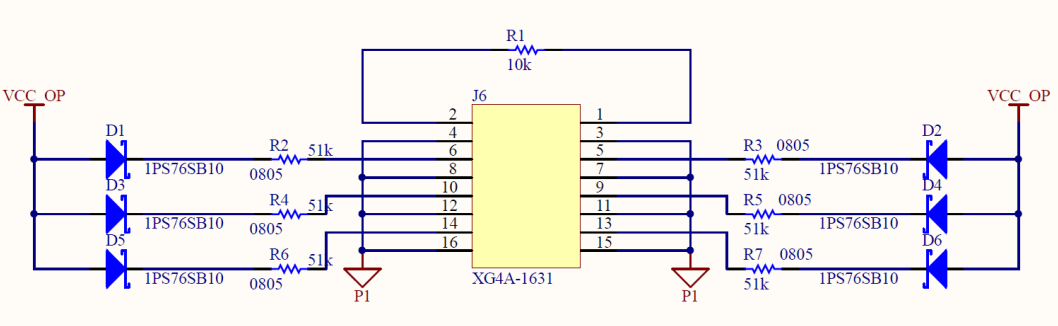


Figura 111 - Connettore J6 scheda IIB.

Il connettore J2 della scheda IIB di Figura 110 permette la trasmissione dei comandi di gamba, del segnale HALT e del segnale di abilitazione della modulazione. Il connettore J6 di Figura 111 permette la trasmissione alla scheda Driver Board di alcuni segnali riservati interni, necessari al funzionamento della scheda stessa per i quali si rimanda al datasheet in bibliografia [12].

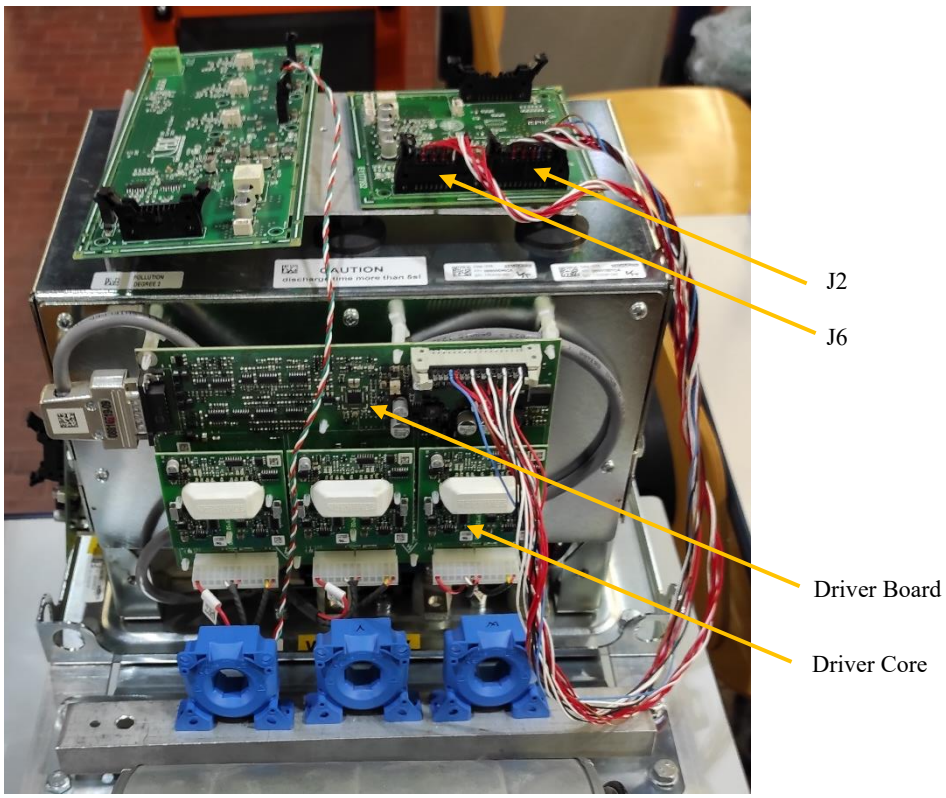


Figura 112 - Connettore per scheda Driver.

5.2.3 Connettori dSPACE

Si prosegue con i connettori di collegamento fra la scheda FPGA e dSPACE, attraverso i quali i due sistemi scambiano i dati.

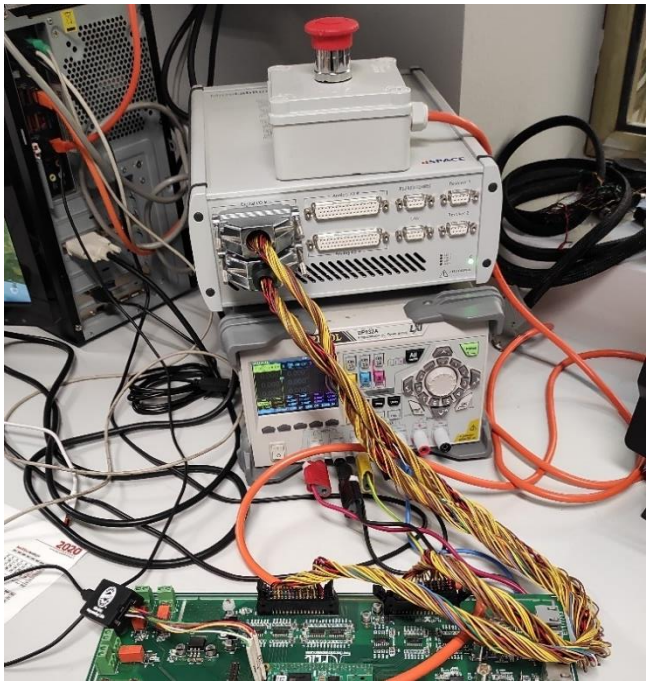


Figura 113 - Connettore per dSPACE

Previa realizzazione è necessario organizzare il pinout degli stessi, in quanto, ad eccezione delle alimentazioni, nessun segnale ha un pin dedicato. In Tabella 3 e in

Tabella 4 si riporta il pinout realizzato per i connettori in questione, nel quale ad ogni segnale viene abbinato un pin del connettore lato FPGA e uno del connettore lato dSPACE.

Tabella 3 - Pinout connettore dSPACE Out FPGA In.

Signal	PinName FPGA	PinNumber FPGA	PinName dSPACE	PinNumber dSPACE
dSPACE_EOP	IN_0	18	DIO1 ch1	17
RD_WR	IN_1	20	DIO1 ch2	16
REQUEST	IN_2	22	DIO1 ch3	15
WORD_IN [0]	IN_3	24	DIO1 ch4	14
WORD_IN [1]	IN_4	26	DIO1 ch5	13
WORD_IN [2]	IN_5	28	DIO1 ch6	12
WORD_IN [3]	IN_6	27	DIO1 ch7	11
WORD_IN [4]	IN_7	25	DIO1 ch8	10
WORD_IN [5]	IN_8	23	DIO1 ch9	9
WORD_IN [6]	IN_9	21	DIO1 ch10	8
WORD_IN [7]	IN_10	19	DIO1 ch11	7
WORD_IN [8]	IN_11	17	DIO1 ch12	6
WORD_IN [9]	IN_12	15	DIO1 ch13	5
WORD_IN [10]	IN_13	13	DIO1 ch14	4
WORD_IN [11]	IN_14	11	DIO1 ch15	3
	IN_15	9	DIO1 ch18	48
	IN_16	7	DIO1 ch19	47
	IN_17	5	DIO1 ch20	46
	IN_18	6	DIO1 ch21	45
	IN_19	8	DIO1 ch22	44
	IN_20	10	DIO1 ch23	43
	IN_21	12	DIO1 ch24	42
	IN_22	14	DIO1 ch25	41
	IN_23	16	DIO1 ch26	40
5V		1	DIO1 ch16	2
5V		2	DIO1 ch17	34
5V		29	DIO1 ch32	49
GND		3	GND	1
GND		4	GND	18
GND		30	GND	50

Tabella 4 - Pinout connettore dSPACE In FPGA Out.

Signal	PinName FPGA	PinNumber FPGA	PinName dSPACE	PinNumber dSPACE
IRQ_dSPACE	OUT_0	16	DIO1 ch33	17
WORD_OUT [0]	OUT_1	14	DIO1 ch34	16
WORD_OUT [1]	OUT_2	12	DIO1 ch35	15
WORD_OUT [2]	OUT_3	10	DIO1 ch36	14
WORD_OUT [3]	OUT_4	8	DIO1 ch37	13
WORD_OUT [4]	OUT_5	6	DIO1 ch38	12
WORD_OUT [5]	OUT_6	5	DIO1 ch39	11
WORD_OUT [6]	OUT_7	7	DIO1 ch40	10
WORD_OUT [7]	OUT_8	9	DIO1 ch41	9
WORD_OUT [8]	OUT_9	11	DIO1 ch42	8
WORD_OUT [9]	OUT_10	13	DIO1 ch43	7
WORD_OUT [10]	OUT_11	15	DIO1 ch44	6
WORD_OUT [11]	OUT_12	17	DIO1 ch45	5
WORD_OUT [12]	OUT_13	19	DIO1 ch46	4
WORD_OUT [13]	OUT_14	21	DIO1 ch47	3
	OUT_15	23	DIO2 ch1	33
	OUT_16	25	DIO2 ch2	32
	OUT_17	27	DIO2 ch4	29
	OUT_18	28	DIO2 ch5	28
	OUT_19	26	DIO2 ch6	27
	OUT_20	24	DIO2 ch7	25
	OUT_21	22	DIO2 ch8	24
	OUT_22	20	DIO2 ch9	23
	OUT_23	18	DIO2 ch10	21
5V		1	DIO1 ch48	2
5V		2	DIO2 ch12	19
5V		29	DIO2 ch3	31
GND		3	GND	1
GND		4	GND	18
GND		30	GND	30

5.3 Testing preliminare schede

Dopo aver assemblato le schede se ne testa il loro funzionamento base, ovvero per ciascuna scheda si esegue una verifica del circuito di alimentazione della stessa e degli eventuali circuiti di condizionamento.

Di ogni scheda si osservano gli schematici forniti per individuare la corretta tensione di alimentazione. Si collega allora la scheda da testare ad un alimentatore bipolare che eroga la tensione opportuna, limitando la corrente ad un valore tale da proteggere i componenti sulla scheda. A questo punto su ciascun test-point dedicato presente sulla scheda si verifica con un multimetro che il livello di tensione sia quello atteso.

Dopo aver applicato questo procedimento per la verifica delle alimentazioni su ogni scheda, di ciascuna se ne testano le funzioni specifiche.

5.3.1 Verifica scheda CAB

Della scheda CAB si verifica che l'acquisizione di corrente effettuata tramite sensore LEM dia lo stesso risultato dell'acquisizione della stessa corrente per mezzo di una pinza amperometrica. Di seguito si riporta il risultato della prova descritta.

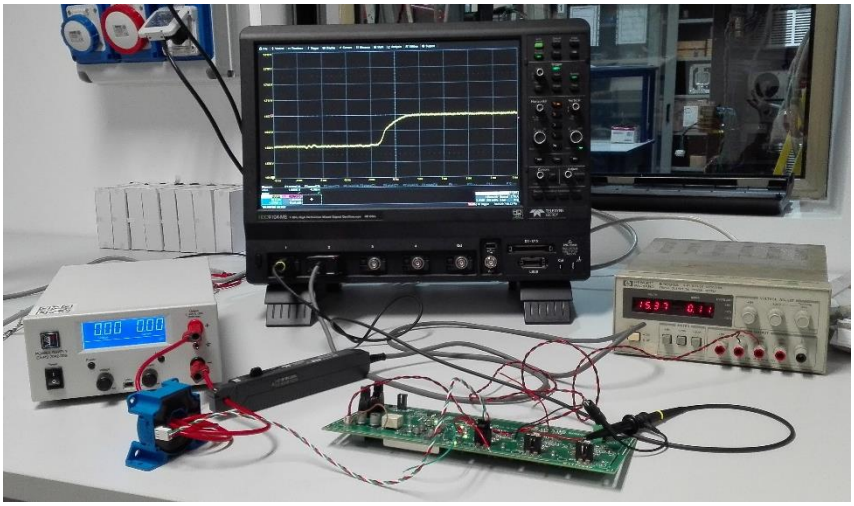


Figura 114 - Setup testing CAB.

In Figura 115 si presenta il risultato dell'acquisizione di una corrente che viene fatta variare a gradino tramite due diversi strumenti. In giallo la corrente in uscita dalla scheda CAB e misurata tramite sensore LEM, mentre in rosso il risultato ottenuto con una pinza amperometrica. Se opportunamente filtrati i due risultati si sovrappongono e quindi l'acquisizione può ritenersi soddisfacente.

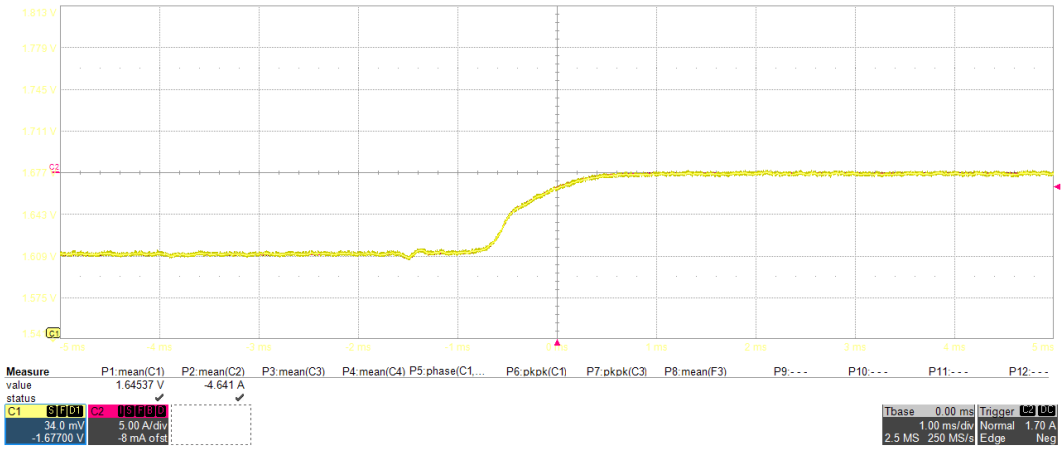


Figura 115 - Verifica dell'acquisizione di corrente tramite scheda CAB.

La stessa scheda ha il compito di misurare anche la tensione di DC-link dell'inverter a cui è collegata. Si alimenta allora il corrispondente canale di acquisizione con un alimentatore di tensione continua, erogando una tensione che

viene fatta variare tra $0\,V$ e $750\,V$, in modo da verificare che questi due valori limite corrispondano rispettivamente allo zero e al fondo scala della scheda, ovvero $3{,}3\,V$.

5.3.2 Verifica scheda VAB

Si verifica ora il corretto funzionamento dei canali di acquisizione della scheda VAB. Dopo la verifica delle alimentazioni si sceglie di alimentare i canali con una tensione sinusoidale e misurare la tensione in ingresso al canale di acquisizione e quella in uscita dal DAC, sfruttando perciò i moduli di gestione implementati sulla scheda FPGA.

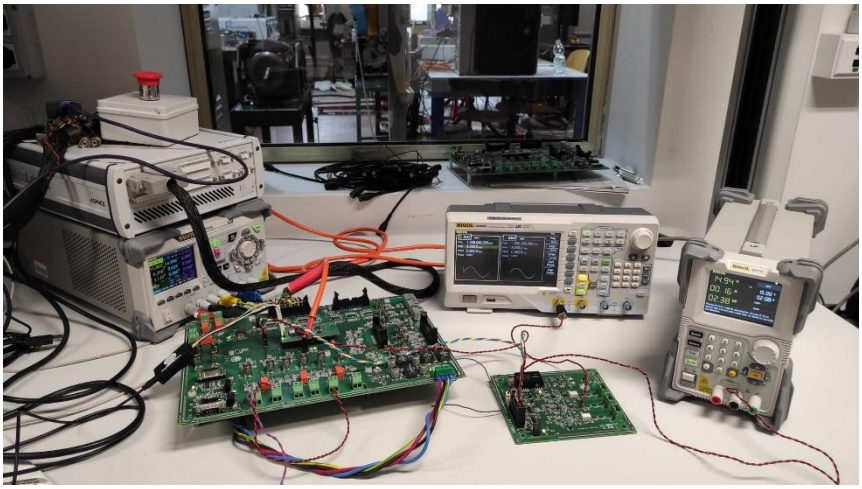


Figura 116 - Setup testing CAB.

Grazie a questa prova, oltre a testare il funzionamento della scheda stessa, si verifica che i moduli di gestione ADC e DAC implementati in Vivado siano validi. La sinusoide prodotta dal generatore di funzioni in Figura 116 viene infatti acquisita dall'ADC della scheda VAB e rimandata in uscita dal DAC. Il confronto, in questo caso, avviene dunque fra la grandezza da acquisire e quella acquisita.

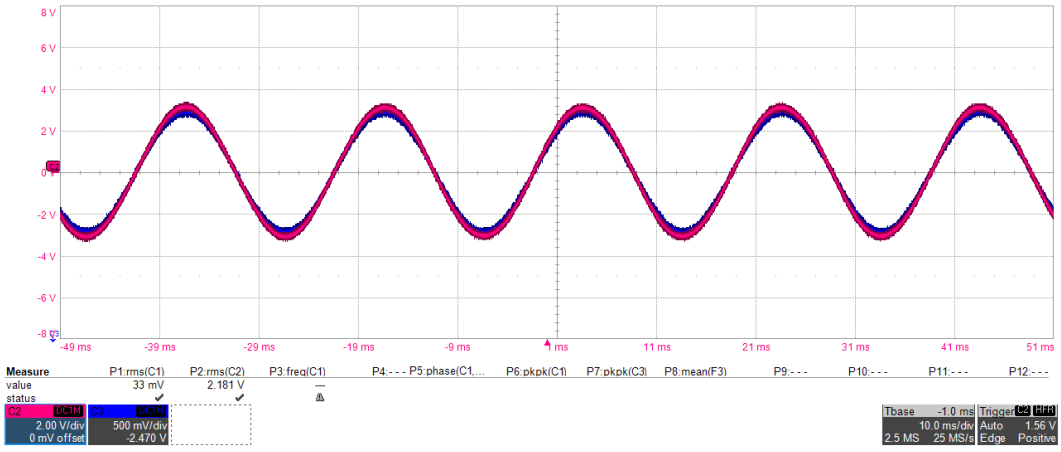


Figura 117 - Verifica acquisizione di tensione tramite scheda VAB.

In Figura 117 è possibile osservare che la tensione acquisita tramite la scheda VAB ed elaborata dal DAC, onda blu in Figura, è del tutto somigliante alla tensione di colore rosso, rappresentante la tensione di alimentazione, a meno di un opportuno fattore di scala.

5.3.3 Verifica scheda IIB

La scheda di interfaccia inverter ha il compito di elaborare i segnali di gamba in arrivo dall'FPGA per produrre quelli definitivi con cui comandare gli switch degli inverter Semikron. In Figura 118 si mostra la generazione dei segnali necessari all'ulteriore circuito integrato mostrato in Figura 120, che produce i comandi di gate definitivi.

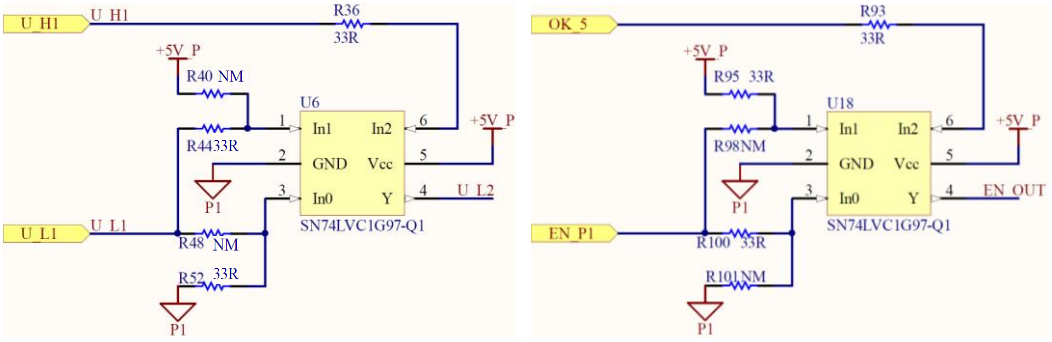


Figura 118 - Generazione dei segnali in ingresso a UCC27537DBVR.

I due circuiti integrati di Figura 118 sono internamente composti da un sistema di porte logiche, mostrato in Figura 119, la cui tavola di verità si riporta in Tabella 5. Il circuito di sinistra, a partire dai comandi alto (In2) e basso (In1) di una determinata fase e il ground (In0), produce in uscita il comando dello switch basso, filtrato da una logica atta ad evitare i cortocircuiti di gamba. Sulla scheda sono presenti un totale di sei di questi circuiti, uno per ogni comando di switch. A destra si presenta invece il circuito che con la stessa logica genera il segnale di abilitazione, in modo che questo valga 1 solo se la scheda è alimentata ($OK\_5 = 1$) e il segnale di abilitazione generato su dSPACE (EN_P1) sia anch'esso alto.

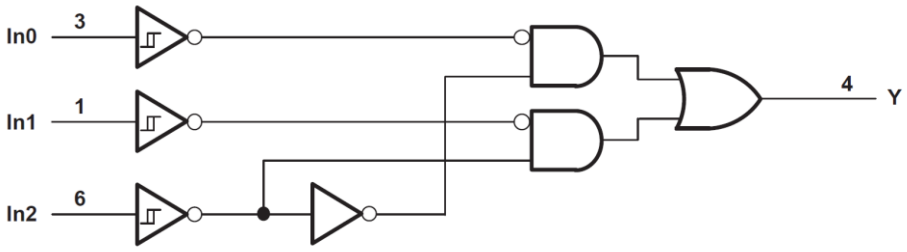


Figura 119 - Circuito logico di SN74LVC1G97-Q1 [9].

Tabella 5 - Tavola di verità di SN74LVC1G97-Q1 [9].

In2	In1	In0	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

In Figura 120 si riporta infine il circuito integrato che ha per uscita il segnale definitivo da mandare ad uno switch dell'inverter. Con logica del tutto analoga al modulo di protezione IGBT implementato in Vivado, questo riproduce l'ingresso in uscita solo se riceve il segnale di abilitazione, altrimenti è sempre nullo.

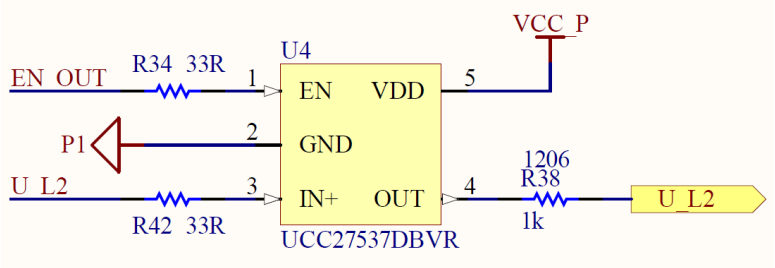


Figura 120 - Generazione del segnale di gamba in uscita da IIB.

Si procede adesso con il testing dei circuiti integrati presentati e la verifica che la logica realizzata sia corretta.

In Figura 121 e Figura 122 si mostra il confronto fra i segnali in ingresso e in uscita dalla scheda IIB. Come è possibile osservare i comandi vengono riprodotti fedelmente, a meno dei ritardi e di un certo fattore di scala, e senza produrre errori.

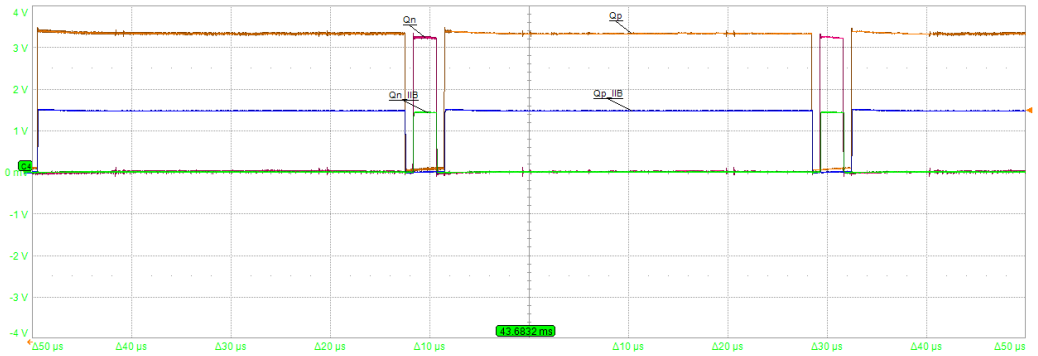


Figura 121 - Confronto segnali di gamba all'uscita dall'FPGA (arancione e rosso) e all'uscita dalla IIB (blu e verde).

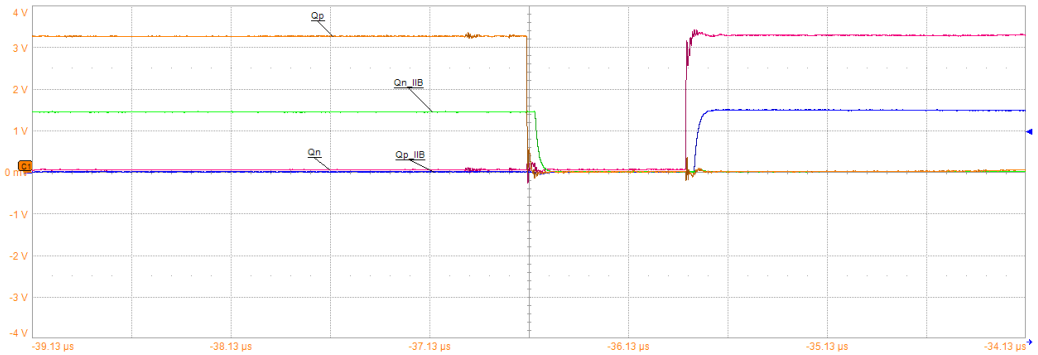


Figura 122 - Ritardi segnali di gamba all'uscita dall'FPGA (arancione e rosso) e all'uscita dalla IIB (blu e verde).

In Figura 123 si verifica il dead time di 800 ns impostato fra il fronte di discesa di un segnale e il fronte di salita del complementare.

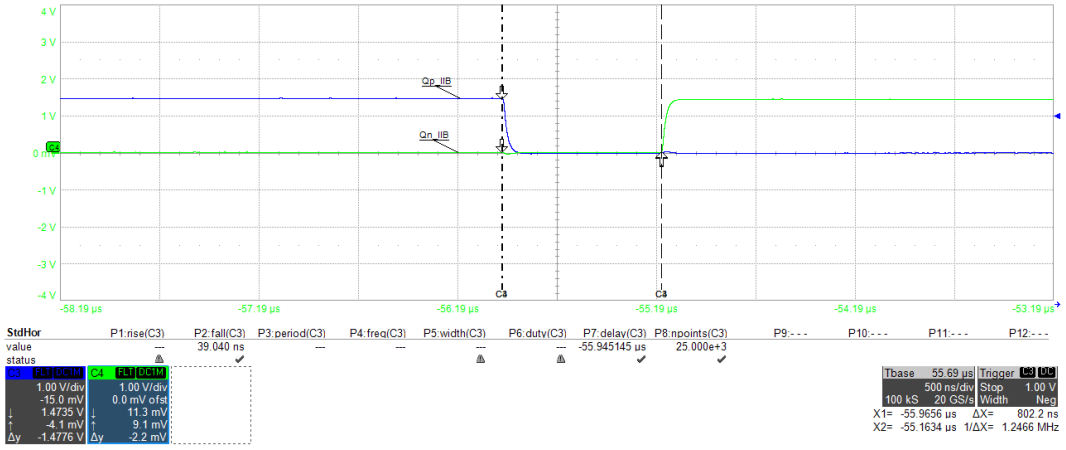


Figura 123 - Verifica del dead time su scheda IIB.

In Figura 124 si mostra inoltre la saturazione a 0 del segnale basso quando il duty cycle della gamba ha un valore tale da non permettere sia la commutazione che la verifica del dead time.

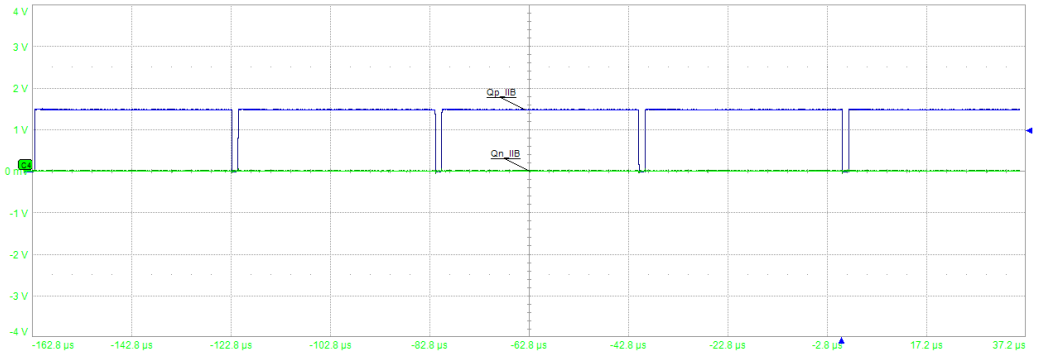


Figura 124 - Verifica saturazione segnali di gamba su scheda IIB.

Si verifica ora il funzionamento dinamico sfruttando l'interfaccia realizzata per dSPACE. Si testa quindi il segnale di abilitazione della modulazione e che i segnali di gamba varino in accordo ai duty cycle impostati manualmente sul ControlDesk.

Come precedentemente esposto, agendo sul pulsante "Progress STATE" il sistema passa, attraversando gli opportuni stati, da quello iniziale di ERROR al READY. In questo stato la modulazione dell'inverter denominato L non è ancora abilitata; ciò avviene solo nello stato START. Con riferimento alla Figura 125 e alla Figura 126 è possibile osservare il passaggio allo stato START per mezzo dell'attivazione dello USER_BUTTON e il conseguente passaggio del led corrispondente al segnale PWM_EN_L dal colore rosso al verde, che indica dunque l'abilitazione della modulazione dell'unità L.

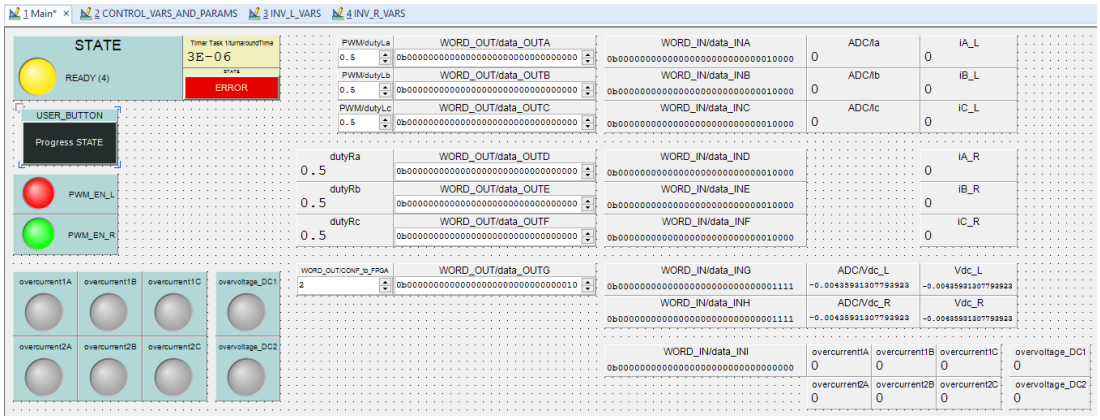


Figura 125 - Main di ControlDesk, stato READY.

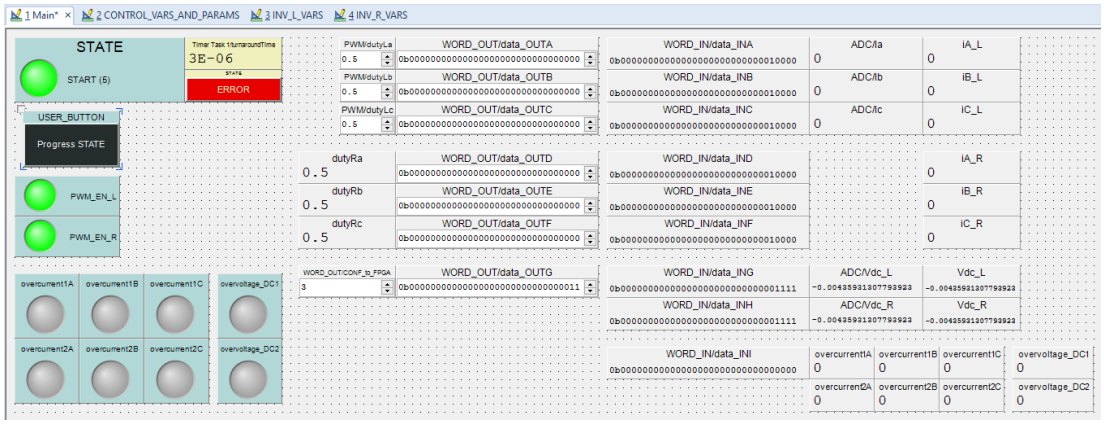


Figura 126 - Main di ControlDesk, stato START.

In Figura 127 si verifica che i comandi di gamba siano diversi da 0 solo in corrispondenza dell'abilitazione della modulazione dell'unità L.

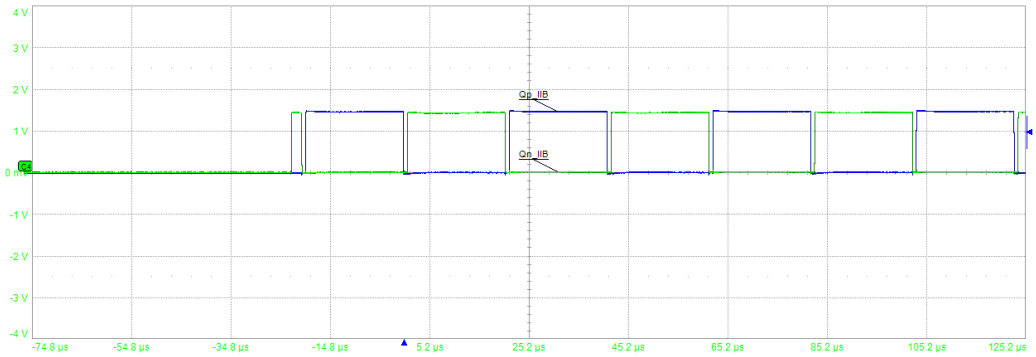


Figura 127 - Segnali di gamba sulla scheda IIB all'ingresso allo stato START.

In Figura 128 è ancora possibile osservare l'effetto della variazione manuale del duty cycle da 0,9 a 0,5 sui segnali di gamba in uscita dalla scheda IIB.

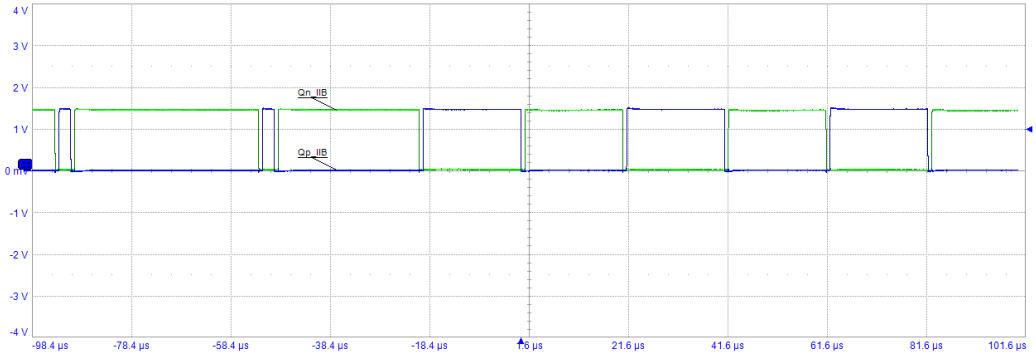


Figura 128 - Effetto della variazione del duty cycle sui segnali di gamba sulla scheda IIB.

Intervenendo sull'apposito pulsante ERROR posto nel ControlDesk si verifica infine che il sistema entri in uno stato sicuro, ovvero che i segnali di gamba alto e basso in uscita dalla scheda IIB si annullino.

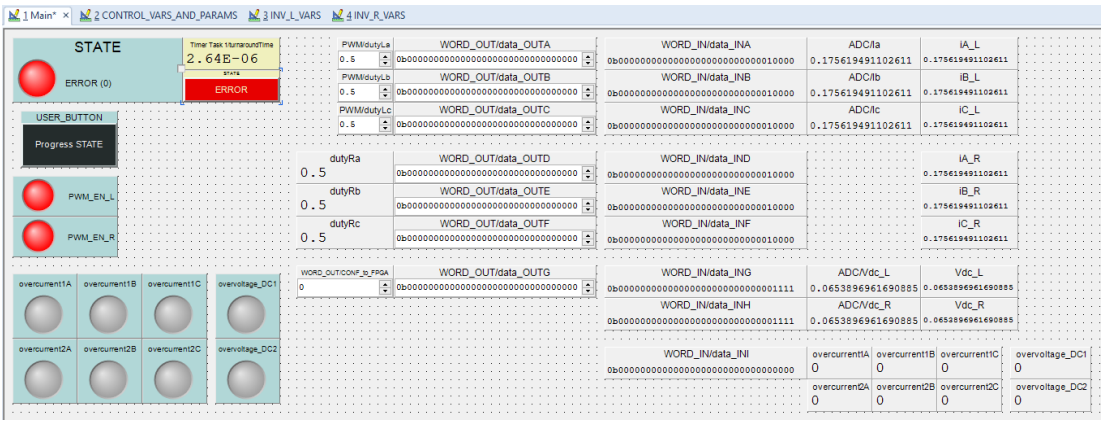


Figura 129 - Main di ControlDesk, stato ERROR.

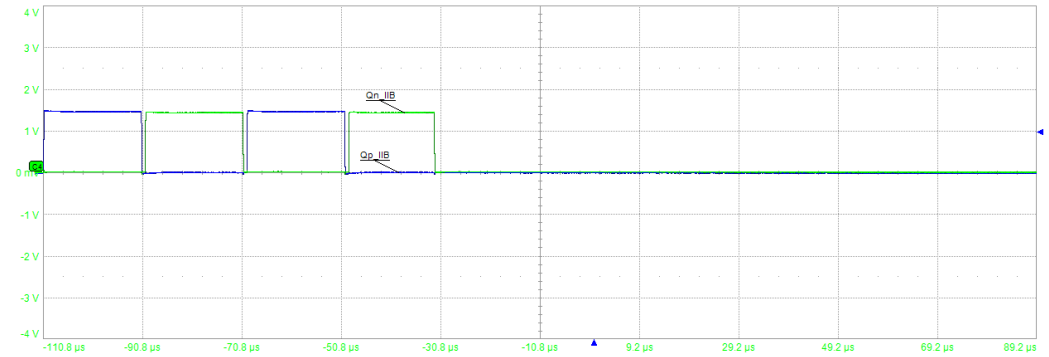


Figura 130 - Segnali di gamba sulla scheda IIB all'ingresso allo stato ERROR.

Si presentano di seguito le forme d'onda prelevate con l'oscilloscopio della scheda FPGA raffiguranti la triangola portante, una modulante variabile e i conseguenti comandi di gamba alto e basso.

Variando la modulante tra il valore massimo e minimo della portante è possibile ostrare un duty cycle variabile tra 100% e 0, passando inoltre per le zone di saturazione mostrate in Figura 132 e in Figura 136.

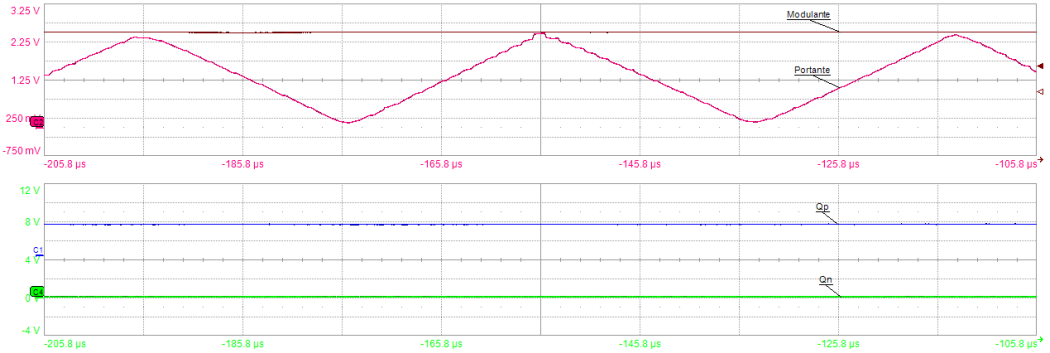


Figura 131 - Modulazione PWM, $d = 100\%$.

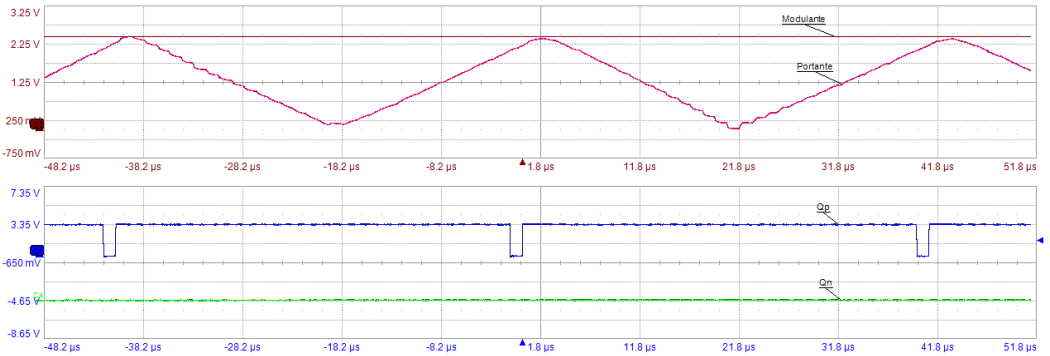


Figura 132 - Modulazione PWM, $d = 95\%$.

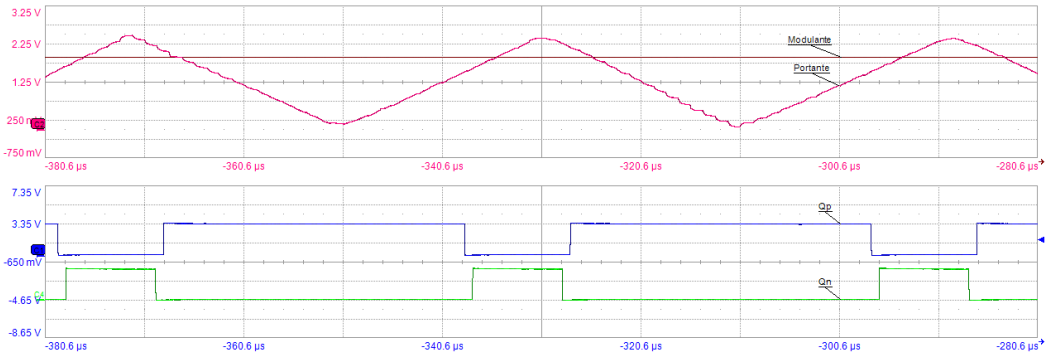


Figura 133 - Modulazione PWM, $d = 75\%$.

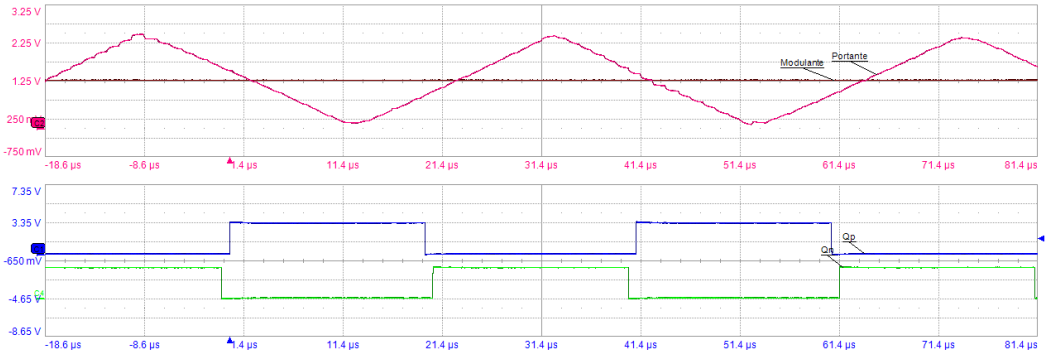


Figura 134 - Modulazione PWM, $d = 50\%$.

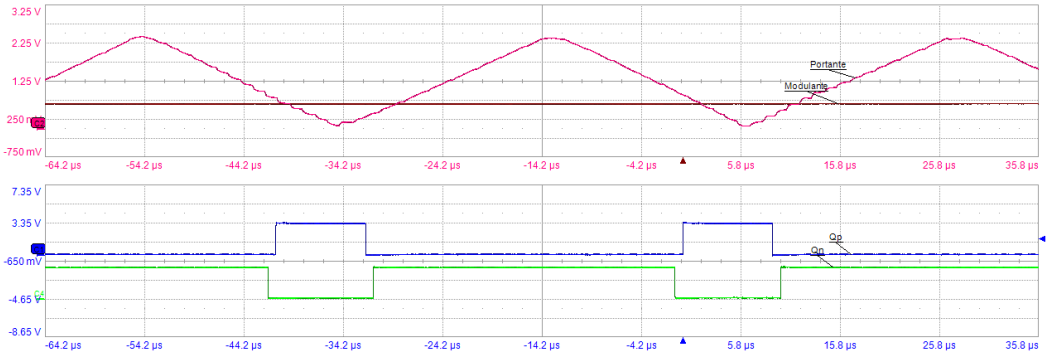


Figura 135 - Modulazione PWM, $d = 25\%$.

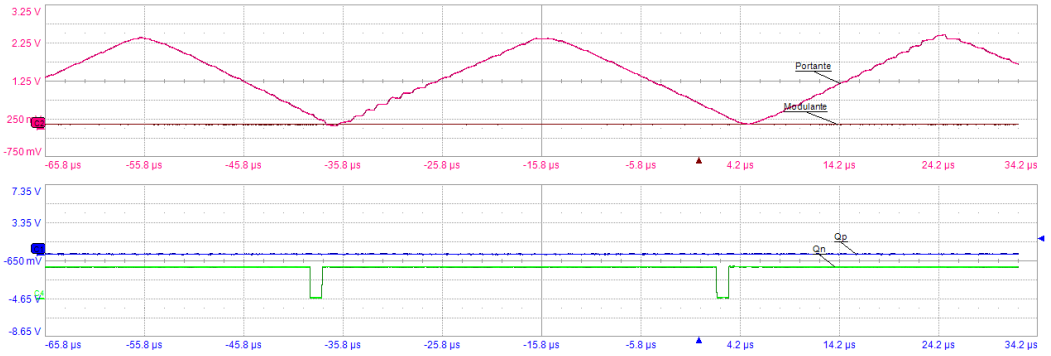


Figura 136 - Modulazione PWM, $d = 5\%$.

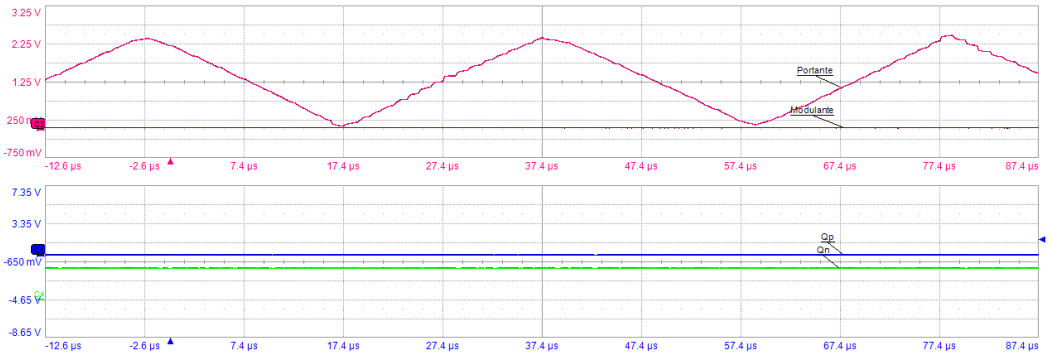


Figura 137 - Modulazione PWM, $d = 0\%$.

5.4 Design supporti

Le schede di misura, così come quella di interfaccia inverter andranno posizionate in prossimità rispettivamente della parte di potenza dello stack e della scheda driver. Si pensa allora di realizzare alcune piastre in alluminio sulle quali montare le suddette schede e di fissare tali piastre allo stack tramite dei supporti ad L.

Per il design di tutte le parti si utilizza il software di disegno e progettazione tridimensionale parametrica SolidWorks, particolarmente utile per la progettazione di apparati meccanici, comunque complessi. A partire dai disegni 3D dello stack e delle PCB delle schede di interesse, si disegnano due piastre per ciascuno stack:

- Un Top Panel, realizzato per essere posizionato sopra il DC-link ed accogliere le schede CAB e IIB;
- Un Side Panel, da mondare al lato del DC-link come sostegno alla scheda VAB.

Le piastre verranno quindi montate sulla copertura del DC-link tramite due profilati ad L, che potranno essere avvitati allo stack sfruttando i fori già presenti senza necessità di crearne di nuovi. Si cercano allora le misure standard commerciali di profili ad L e si realizza il disegno 3D dei supporti ideati, comprendenti anche i fori previsti per il montaggio finale. Con la funzione “Make Drawing from Part” di Solidworks si realizza infine il disegno 2D quotato, che si riporta in Figura 138.

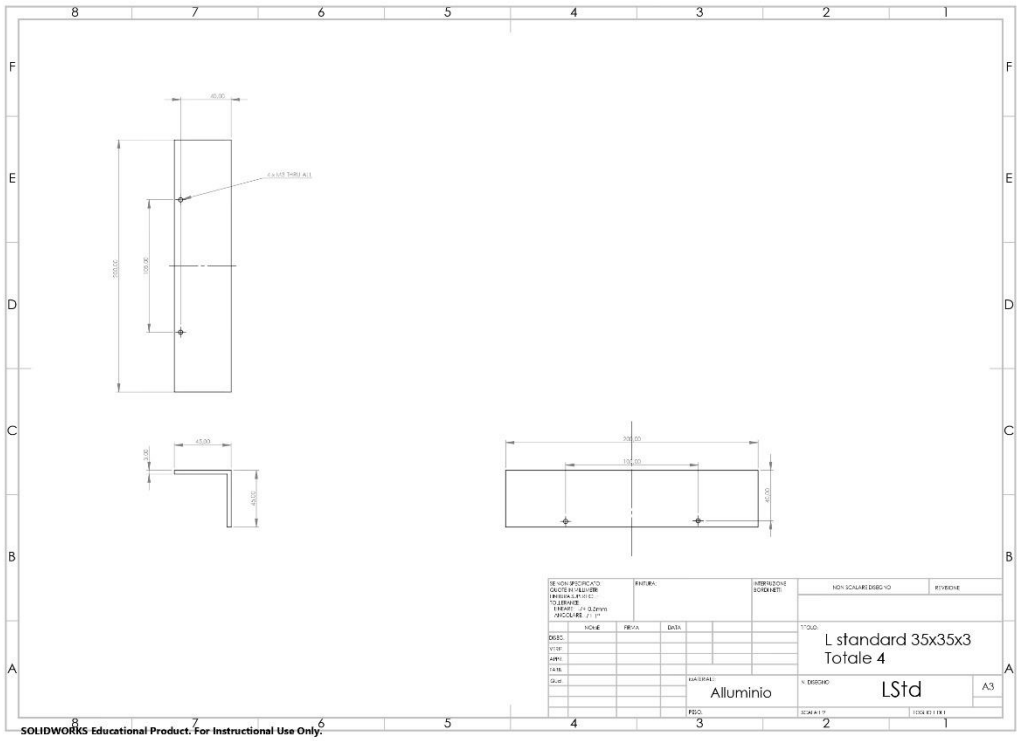
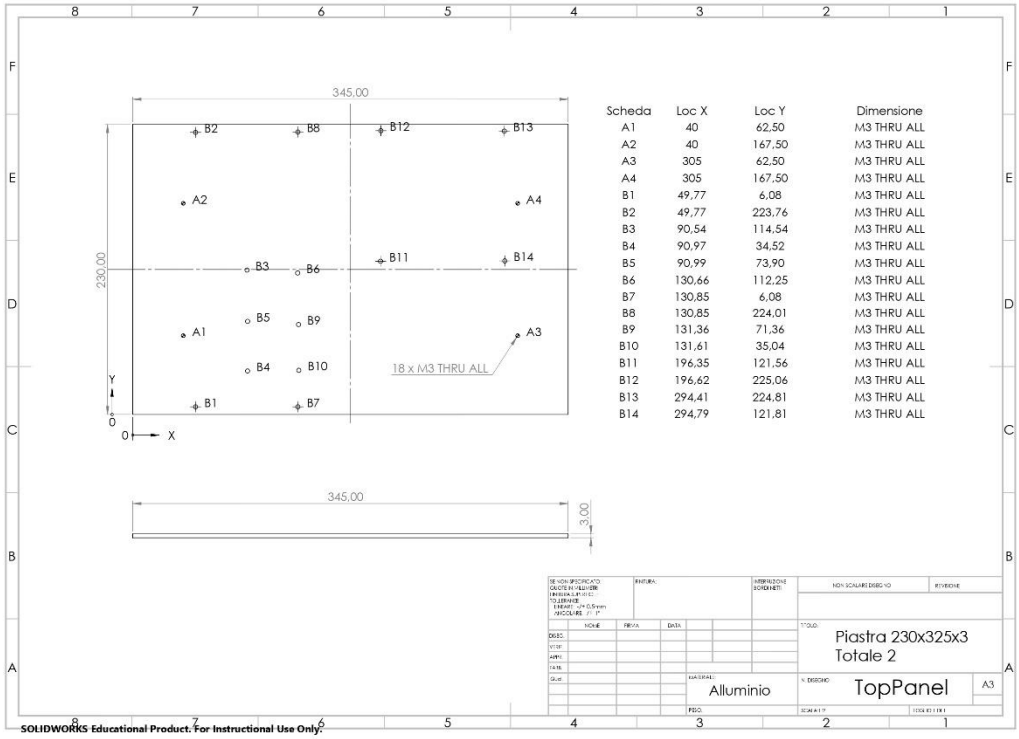


Figura 138 - LStd Solidworks.

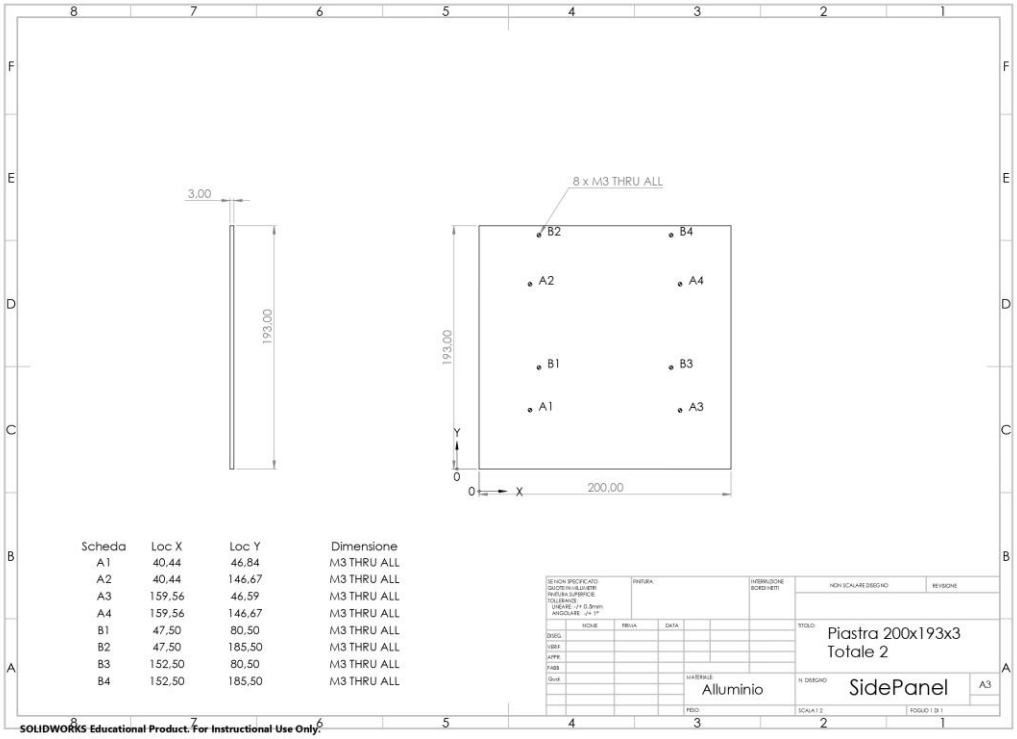
Si prosegue il design dei supporti con la realizzazione del Top Panel. Su questo verranno montate le schede CAB e IIB in modo che la prima risulti avere il connettore per la misura della tensione del DC-link rivolta verso i morsetti di questo e la seconda i connettori J2 e J6, dei quali si è spiegata la funzione nel Paragrafo 5.2.2, verso il connettore della scheda driver dell'inverter. Una volta chiaro il posizionamento relativo dei componenti fra loro si realizzano, anche per la piastra superiore, i disegni 3D e 2D quotato, indicando le dimensioni del pannello di alluminio e la posizione dei fori per l'assemblaggio della piastra ai profilati e delle schede sulla piastra.



Scheda	Loc X	Loc Y	Dimensione
A1	40	62,50	M3 THRU ALL
A2	40	167,50	M3 THRU ALL
A3	305	62,50	M3 THRU ALL
A4	305	167,50	M3 THRU ALL
B1	49,77	6,08	M3 THRU ALL
B2	49,77	223,76	M3 THRU ALL
B3	90,54	114,54	M3 THRU ALL
B4	90,97	34,52	M3 THRU ALL
B5	90,99	73,90	M3 THRU ALL
B6	130,66	112,25	M3 THRU ALL
B7	130,85	6,08	M3 THRU ALL
B8	130,85	224,01	M3 THRU ALL
B9	131,36	71,36	M3 THRU ALL
B10	131,61	35,04	M3 THRU ALL
B11	196,35	121,56	M3 THRU ALL
B12	196,62	225,06	M3 THRU ALL
B13	294,41	224,81	M3 THRU ALL
B14	294,79	121,81	M3 THRU ALL

Figura 139 - TopPanel Solidworks.

Analogamente si realizzano i disegni del pannello laterale, che accoglierà la scheda VAB rivolgendone i connettori per l'acquisizione delle tensioni verso la parte di potenza dello stack.



Scheda	Loc X	Loc Y	Dimensione
A1	40,44	46,84	M3 THRU ALL
A2	40,44	146,67	M3 THRU ALL
A3	159,56	46,59	M3 THRU ALL
A4	159,56	146,67	M3 THRU ALL
B1	47,50	80,50	M3 THRU ALL
B2	47,50	185,50	M3 THRU ALL
B3	152,50	80,50	M3 THRU ALL
B4	152,50	185,50	M3 THRU ALL

Figura 140 - SidePanel Solidworks.

Un ultimo pezzo viene poi realizzato per alloggiare i sensori LEM per la misura delle correnti. Facendo riferimento al datasheet dei sensori [10], nel quale

sono riportate le dimensioni degli stessi e i diametri dei fori, si realizza il disegno di un'altra piastra da posizionare davanti ai terminali di potenza dell'inverter trifase.

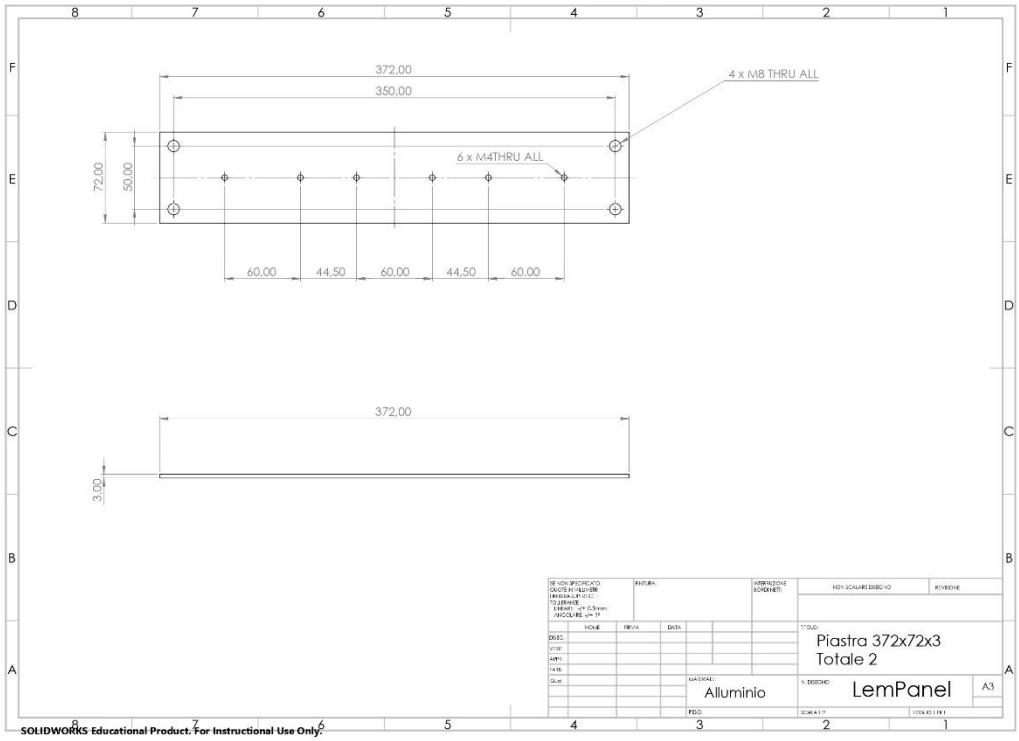


Figura 141 - LemPanel Solidworks.

Su Solidworks si crea dunque un assieme di tutte le parti realizzate, delle PCB e dello stack, in modo da verificare la bontà del progetto e della realizzazione dei supporti. In Figura 142 si presenta l'assieme di tutte le parti su Solidworks.

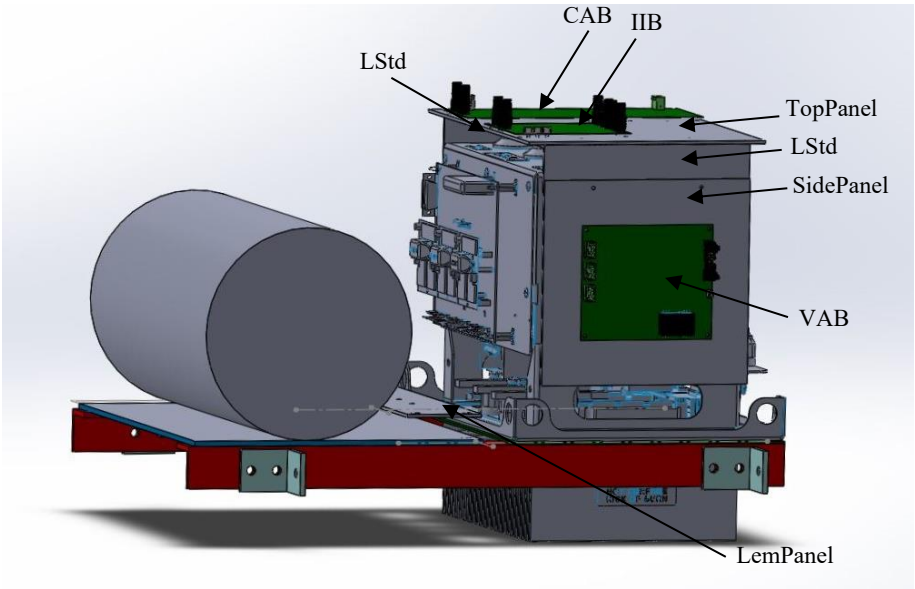


Figura 142 - Sistema stack con schede Solidworks.

Si mostra inoltre una fotografia del sistema fisico reale, con la quale si desidera dare un'idea del montaggio delle PCB e dei sensori su ciascuno stack Semikron.

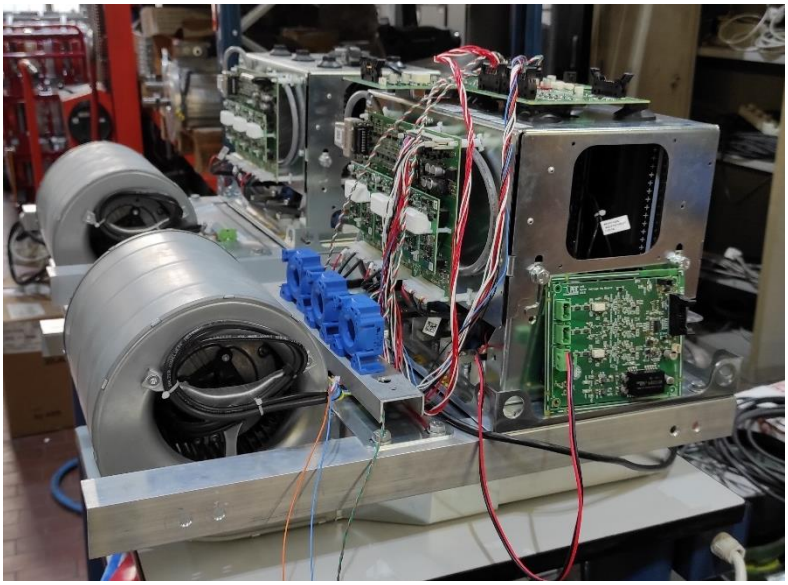


Figura 143 - Sistema stack con schede reale.

5.5 Controllo ventilatore di raffreddamento

Come anticipato al Paragrafo 1.1 gli stack Semikron dispongono di un sistema di raffreddamento realizzato con ventilatori assiali, di cui si riporta il riferimento al datasheet in bibliografia [13]. Questi richiedono un'alimentazione con tensione monofase di 230 V_{AC} e permettono regolare il periodo del loro funzionamento con una modulazione PWM. In prossimità del ventilatore è posta una morsettiera, mostrata in Figura 145, che ne consente l'alimentazione e l'adduzione dei segnali di controllo per la regolazione. Tali segnali sono:

- l'alimentazione del circuito di controllo di $10V_{DC}$, generata internamente
- il segnale di comando;
- il ground.

Si progetta allora un circuito di controllo di cui si riporta lo schema elettrico in Figura 144. Il cuore di tale circuito è l'optoisolatore [16], che consente di isolare il ground del circuito del ventilatore da quello della scheda FPGA. Questo presenta un led al primario ed un fototransistore al secondario, il quale è normalmente aperto, ma viene chiuso se irradiato dal led. Si alimenta quindi il led con il segnale di comando generato dall'FPGA sulla base del duty cycle impostato dal ControlDesk di dSPACE. Quando tale segnale è basso il morsetto PWM è collegato all'alimentazione e il ventilatore entra in funzione, quando il segnale è basso invece il transistor si chiude, cortocircuitando il morsetto PWM a GND, così da arrestare il ventilatore.

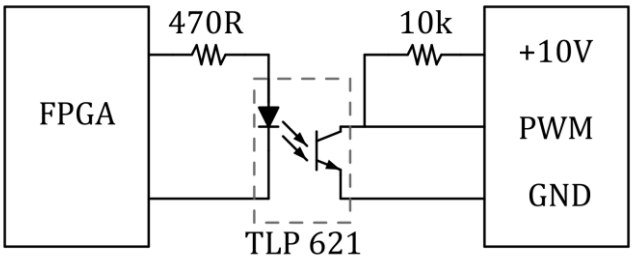


Figura 144 - Schema elettrico del circuito di controllo della ventola.

Si assembla allora su una scheda millefori il circuito, la cui realizzazione fisica è mostrata in Figura 145.

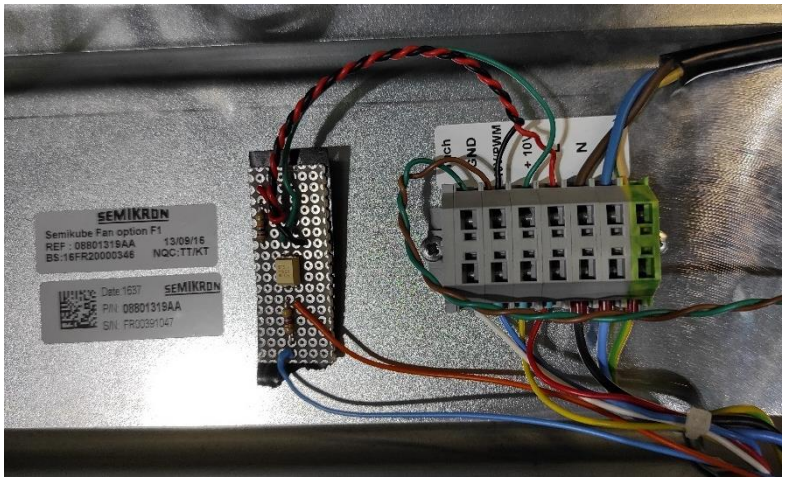


Figura 145 - Circuito di controllo della ventola a destra, morsettiera a sinistra.

Per mezzo dei cavi blu e arancione in Figura si adduce il segnale di comando generato dall'FPGA. In Figura 146 si presentano le forme d'onda dei segnali portante e modulante e il comando prodotto per il ventilatore rilevati con un oscilloscopio.

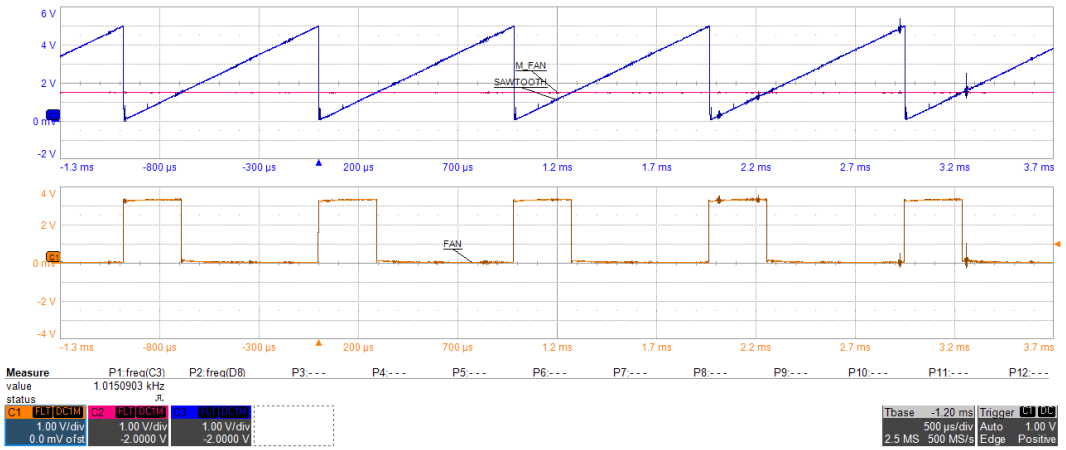


Figura 146 - Segnale di comando della ventola.

Capitolo 6

Conclusioni

L'obiettivo di questa è tesi è stato quello di realizzare un banco prova per il testing dei diodi presenti nei moduli di potenza dei due stack Semikron che si sono presentati al Capitolo 1, controllandoli grazie ad un sistema che impiegasse una scheda FPGA, interfacciata ad alcune schede di misura ed interfaccia, e la piattaforma dSPACE. La scheda FPGA è stata resa responsabile della gestione del codice di basso livello, contenente in particolare i blocchi per la modulazione PWM dei due inverter e il modulo di gestione degli ADC delle schede di misura. Per la descrizione del sistema completo si rimanda al Capitolo 2 e alla Figura 62. L'utilizzo di dSPACE ha invece permesso di condurre prove real-time sul sistema, implementando su questa piattaforma il codice di controllo di alto livello degli inverter, che è stato precedentemente realizzato e testato sul software di simulazione PLECS. Per i dettagli di implementazione del controllo si faccia riferimento al Capitolo 3 e alla Figura 63. Grazie all'interfaccia grafica a PC di dSPACE, infatti, è stato possibile visualizzare su schermo l'andamento delle grandezze desiderate e modificare in tempo reale i parametri dell'applicazione, come descritto al Capitolo 4.

I miei contributi allo sviluppo del progetto sono stati:

- Scrittura firmware FPGA;
- Design e simulazione del controllo su PLECS;
- Scrittura codice dSPACE e realizzazione dell'interfaccia utente;
- Assemblaggio e testing delle schede di controllo, misura ed interfaccia, realizzazione degli appositi connettori e design di alcuni supporti meccanici.

Si prevedono già alcuni sviluppi ed utilizzi futuri per il sistema realizzato. Questo verrà infatti innanzitutto sfruttato per il testing di componenti elettronici, così per come si presenta adesso, ma anche per la caratterizzazione di motori elettrici, con l'aggiunta di schede per l'interfacciamento di encoder, delle quali si era fatto cenno al Paragrafo 1.2.

Bibliografia

[1] AXICOM, T. (s.d.). K1 IM01TS. *Datasheet*.

[2] CYPRESS. (s.d.). S25FL256S. *Datasheet*.

[3] DEVICES, A. (s.d.). ADuM130E. *Digital Isolators*.

[4] DEVICES, A. (s.d.). ADuM160N. *Digital Isolators*.

[5] dSPACE. (2015). RTLib Reference. *MicroLabBox*.

[6] electronic, t. (s.d.). TE020 TRM. *Datasheet*.

[7] Emanuele Giamello, G. P. (2017). *Relazione attività di ricerca AVALON-Cluster "Aerospazio"*.

[8] INSTRUMENTS, T. (2016, April). Micro Power Quad Digital-to-Analog. *DAC124S085*.

[9] INSTRUMENTS, T. (s.d.). SN74LVC1G97-Q1. *Configurable Multiple-Function Gate*.

[10] LEM. (s.d.). Current Transducer LF 305-s.

[11] SEMIKRON. (s.d.). IGDD6-1-426-D1616-E1N6-DL-FA. *3-phase input rectifier + 3-phase inverter*.

[12] SEMIKRON. (s.d.). SEMIKUBE Board GD 11. *IGBT Driver Board*.

[13] SEMIKRON. (s.d.). SKF 16P-230-01. *Radial fan*.

[14] SEMIKRON. (s.d.). SKYPER 32 PRO R. *IGBT Driver Core*.

[15] TECHNOLOGY, L. (s.d.). Sampling ADC in TSOT. *LTC2314-14*.

[16] TOSHIBA. (s.d.). TLP621. *TOSHIBA Photocouplet*.

[17] XILINX. (2016, October 3). DS187 (v1.19). *Artix-7 FPGAs Data Sheet*.

[18] XILINX. (2018, July 30). UG472 (V1.14). *7 Series FPGAs Clocking Resources*.