UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

TESI DI LAUREA MAGISTRALE

MISURA DELLA TENSIONE IN INVERTER TRIFASE TRAMITE FPGA

Relatore: Ch.mo Prof. MAURO ZIGLIOTTO

Laureando: PAOLO GRAFFAGNINO

Matricola 566919-IL

ANNO ACCADEMICO 2009-2010

Sommario

Le effettive tensioni di alimentazione di un motore elettrico fornite da un inverter trifase PWM, non sono quelle ideali generate dagli algoritmi di controllo. Infatti l'inverter introduce delle non linearità causate dall'utilizzo di interruttori elettronici di potenza (che hanno delle caratteristiche non lineari e dei tempi di accensione e spegnimento non istantanei). Inoltre, per evitare il cortocircuito sull'alimentazione, vengono appositamente introdotti dei tempi morti, causa anch'essi di non idealità nelle tensioni di alimentazione. Pertanto risulta importante la conoscenza delle effettive tensioni di alimentazione del motore.

In questo progetto si propone la realizzazione di un sistema di acquisizione digitale delle tensioni concatenate, fornite da un inverter trifase con modulazione PWM. In particolare si tratteranno la realizzazione hardware del sistema di acquisizione, e la creazione del software di elaborazione. Il cuore del progetto è composto da una scheda FPGA, la quale riceve in ingresso i valori digitali delle tensioni concatenate e produce in uscita i valori digitali delle fondamentali delle tensioni concatenate stesse. Le tensioni digitali ottenute, saranno utilizzate in fasi successive al progetto stesso, all'interno di algoritmi sensorless per la stima di coppia.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

INTRODUZIONE E OBIETTIVI DEL PROGETTO

1.1 Topologia dell'Inverter trifase

1.1.1 Introduzione all'Inverter e Modulazione PWM

I convertitori dc-ac sono detti inverter. La funzione di un inverter è quella di trasformare una tensione continua d'ingresso in una tensione di uscita alternata, simmetrica e di ampiezza e frequenza prefissate. Se la tensione di ingresso è fissa e non controllabile, una tensione di uscita variabile può essere ottenuta variando il guadagno dell'inverter e questo lo si può ottenere per mezzo della tecnica PWM (Pulse Width Modulation o modulazione a larghezza d'impulso). Il guadagno può essere definito come il rapporto tra la tensione alternata di uscita e la tensione continua d'ingresso. La media mobile (sul periodo PWM) della tensione di uscita di un inverter ideale dovrebbe essere sinusoidale. Tuttavia le forme d'onda degli inverter effettivamente realizzati non sono sinusoidali e contengono armoniche. Per applicazioni a bassa e media potenza possono essere accettabili tensioni a onda quadra o quasi quadra, mentre per applicazioni ad alta potenza sono richieste forme d'onda sinusoidale a bassa distorsione. Con la disponibilità di dispositivi di potenza a semiconduttore ad alta velocità, il contenuto armonico della tensione di uscita può essere ridotto significativamente mediante tecniche di modulazione. Gli inverter sono largamente utilizzati in applicazioni industriali, per esempio: motori in alternata a velocità variabile, riscaldamento ad induzione, gruppi di continuità, ecc... L'ingresso può essere costituito da una batteria, una cella a combustibile, un pannello solare e altre sorgenti in continua. Le tipiche uscite per sistemi trifase ad alta potenza sono: 220V-380V a 50Hz, 120V-208V a 60Hz e 115V-200V a 400Hz. Per ottenere un sistema trifase di tensione, si possono riunire tre inverter monofase a mezzo ponte, modulati in PWM, come mostrato in Fig. 1.1. I tre inverter hanno in comune una stessa alimentazione che si suppone costituita da due sorgenti di tensione di ampiezza $E/2$. Ognuno dei tre inverter monofase comanda una fase dell'inverter trifase. Le gambe delle tre fasi 1, 2, 3, comprendono rispettivamente gli interruttori S_{1p} e S_{1n}, S_{2p} e S_{2n}, S_{3p} e S_{3n}.

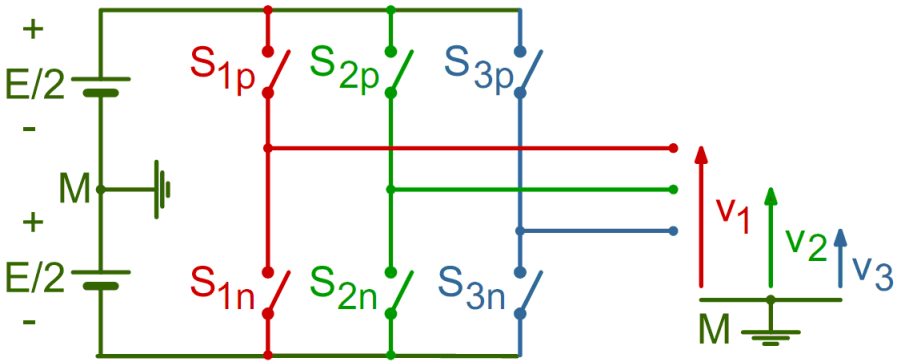

Fig. 1.1: Inverter Trifase Ideale (Fig. ricavata da [1])

Ciascuna gamba può essere modulata in modo indipendente dalle altre, in modo da seguire una sua modulante di riferimento. Solitamente però, per le tre gambe si usa

la stessa frequenza di modulazione e spesso si usa la stessa portante triangolare. Le tre modulanti sono scelte in modo da dare il sistema trifase voluto, che in molti casi è sinusoidale e simmetrico, con tensioni di uguale ampiezza e sfasate tra loro di 120°.

In Fig. 1.2 è mostrato il procedimento di modulazione analogica trifase a PWM, con tre modulanti sinusoidali v_1^*, v_2^*, v_3^* di frequenza ed ampiezza uguali, con la stessa portante triangolare per le tre fasi. Sono anche mostrate le tensioni istantanee v_1, v_2, v_3 uscenti da ciascuna gamba, e gli andamenti delle tensioni medie v_{1med}, v_{2med}, v_{3med} nel periodo di modulazione. Le tensioni sono riferite al punto M intermedio alle due alimentazioni.

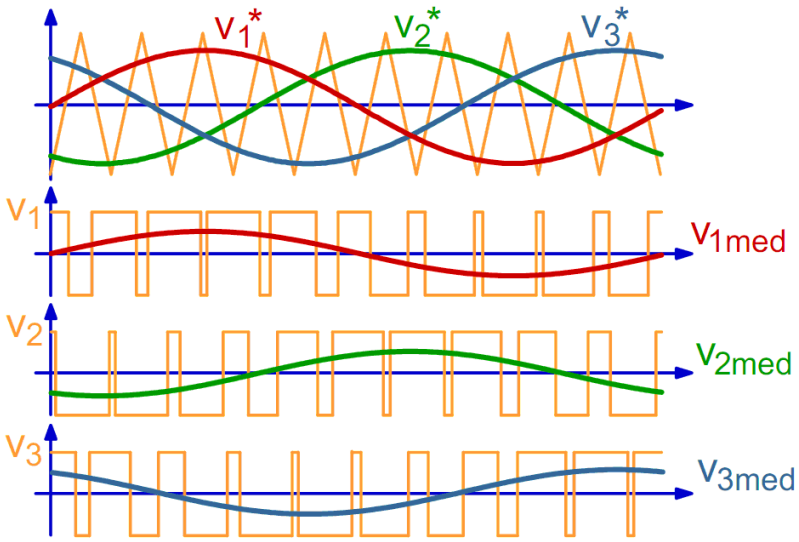

Fig. 1.2: Generazione di tre tensioni sinusoidali simmetriche con portante triangolare comune (Fig. ricavata da [1])

Per ciascuna delle tensioni medie di fase v_{1med}, v_{2med}, v_{3med} vale quanto segue. Definito m_a come il rapporto $m_a = V^*_{max}/V_{trmax}$ con V^*_{max} valore massimo della modulante e V_{trmax} valore massimo della portante triangolare. Si ottiene il valore massimo V_{medmax} delle tensioni di fase uguale a $V_{medmax} = m_a * E/2$. Ciò vale se non si ha sovramodulazione, cioè se $m_a \leq 1$.

Nel sistema trifase si possono definire le tensioni concatenate $v_{12} = v_1 - v_2$, $v_{23} = v_2 - v_3$, $v_{31} = v_3 - v_1$. Dalle corrispondenti tensioni medie si hanno le tensioni medie concatenate $v_{12med} = v_{1med} - v_{2med}$, $v_{23med} = v_{2med} - v_{3med}$, $v_{31med} = v_{3med} - v_{1med}$. Poiché le tensioni medie di fase sono, almeno teoricamente, sinusoidali e simmetriche, anche le tensioni concatenate lo sono e il loro valore massimo di fase $V_{concmax}$, sempre in assenza di sovramodulazione, vale:

$$V_{concmax} = v_{12medmax} = v_{23medmax} = v_{31medmax} = \sqrt{3}V_{medmax} = \sqrt{3}m_a\frac{E}{2} \tag{1.1}$$

Il corrispondente valore efficace V_{eff} della tensione concatenata prodotta dall'invertitore trifase è quindi:

$$V_{eff} = \frac{V_{concmax}}{\sqrt{2}} = \frac{\sqrt{3}}{\sqrt{2}}V_{medmax} = \frac{\sqrt{3}}{2\sqrt{2}}m_aE = 0,612m_aE \Leftrightarrow (m_a \leq 1) \tag{1.2}$$

1.1.2 Implementazione dell'inverter

L'implementazione dell'inverter monofase a mezzo ponte (i ragionamenti sono del tutto analoghi per l'inverter monofase a ponte intero e per l'inverter trifase) cioè la sua realiz-

zazione pratica, si fa sostituendo a ciascuno degli interruttori ideali S_1 e S_2 dello schema uno dei componenti attivi di potenza, adatti ad operare in commutazione e capaci di essere comandati sia in accensione sia in spegnimento. Si possono così usare MOSFET, transistori bipolari, IGBT, GTO o anche altri componenti capaci di commutazione comandata. Poiché, la corrente deve poter fluire negli interruttori in entrambe le direzioni, mentre la tensione ai capi degli interruttori non diventa mai negativa, i componenti con diodo intrinseco sono adatti a sostituire gli interruttori ideali in tale tipo di convertitore. Per gli altri componenti a 'blocco inverso', la capacità di conduzione inversa deve essere ottenuta connettendo in antiparallelo un diodo. Si tratta di un componente distinto dal dispositivo di commutazione vero e proprio, anche se spesso esso viene racchiuso nello stesso involucro del componente attivo. Le sorgenti di alimentazione continua del convertitore, supposte nel caso ideale come generatori $E/2$ privi di impedenza, presentano in pratica cadute di tensione causate dalla corrente che si possono rappresentare come impedenze interne Z_s (Fig. 1.3). La corrente erogata dai generatori di alimentazione è molto variabile, con fronti ripidi dovuti alle commutazioni degli interruttori. Le impedenze interne Z_s, specie se di valore elevato alle alte frequenze, causano quindi cadute di tensione che a loro volta provocano serie distorsioni alla forma d'onda generata dal convertitore. Per offrire una via a bassa impedenza, alle componenti ad alta frequenza della corrente di alimentazione, si usano di regola, dei condensatori in parallelo ai terminali della alimentazione, (C_1, C_2 in Fig. 1.3). Più avanti, per semplicità, tali condensatori saranno sottintesi e verranno indicati i generatori ideali. Visto l'elevato valore delle capacità richiesto, normalmente tali condensatori sono del tipo elettrolitico, con eventualmente in parallelo condensatori più piccoli di tipo poliestere (per filtrare meglio le componenti ad alta frequenza).

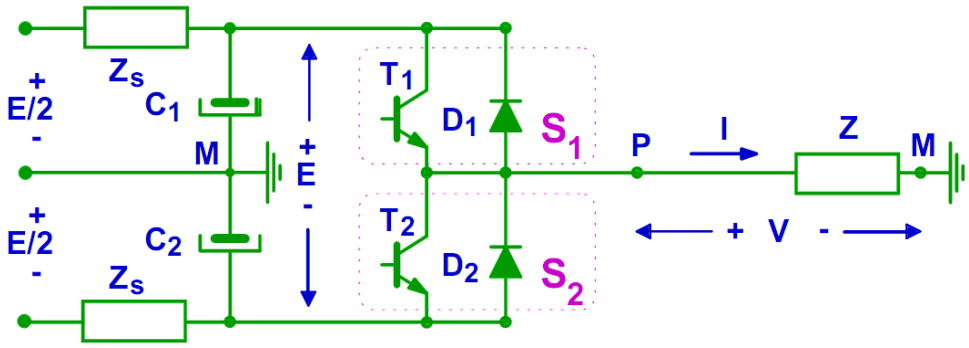

Fig. 1.3: Schema Realizzativo (Fig. ricavata da [1])

1.1.3 Commutazioni e tempi morti

Tutti i componenti attivi di commutazione, sia pure in diversa misura, hanno la caratteristica che le commutazioni della corrente i_s avvengono con un certo ritardo rispetto ai segnali di comando se applicati all'elettrodo di controllo. Si ha così (Fig. 1.4) un ritardo di accensione t_{don}, (ritardo con cui inizia la conduzione) ed un tempo di salita t_{ron} fino alla conduzione completa. Similmente, si ha un tempo di ritardo t_{doff} prima che inizi lo spegnimento ed un tempo di discesa t_{foff} fino alla completa interdizione . I segnali di comando S_C delle commutazioni del convertitore vengono di solito generati a 'basso livello', in appositi circuiti analogici e/o digitali, ed amplificati e trasmessi (spesso con isolamento) agli elettrodi di comando dei dispositivi attivi. Gli stadi di amplificazione, di trasmissione e di isolamento introducono ulteriori ritardi, t_{con} all'accensione e t_{coff} allo spegnimento, che si aggiungono a quelli propri dei componenti di potenza. Tra il segnale di comando S_C all'accensione e l'inizio effettivo della conduzione si ha dunque un ritardo complessivo

$t_{ton} = t_{don} + t_{con}$. Invece, allo spegnimento, tra il fronte del segnale di comando S_C e l'interdizione completa si ha un ritardo complessivo $t_{toff} = t_{doff} + t_{foff} + t_{coff}$.

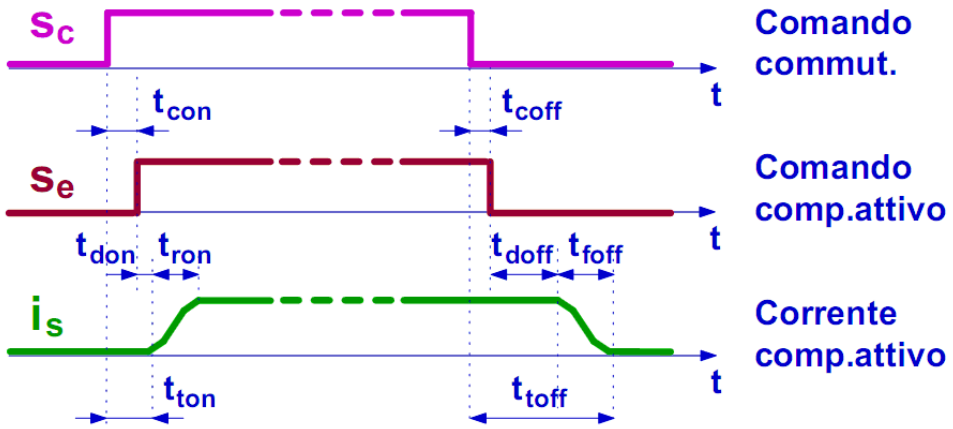

Fig. 1.4: Ritardi di commutazione (Fig. ricavata da [1])

Per un dato componente, in condizioni e/o in tempi diversi, e tra componenti diversi, anche dello stesso tipo, vi sono incertezze e variazioni sia nei ritardi di trasmissione t_{con}, t_{coff} sia nei ritardi e nei tempi di commutazione dei dispositivi attivi t_{don}, t_{doff}, t_{foff}. Analoghe incertezze si hanno, di conseguenza, nella valutazione dei ritardi totali t_{ton}, t_{toff}. Per tener conto di tali incertezze e della loro entità. Si possono definire (Fig. 1.5) valori massimi t_{maxon}, t_{maxoff} e valori minimi t_{minon}, t_{minoff} dei ritardi totali stessi.

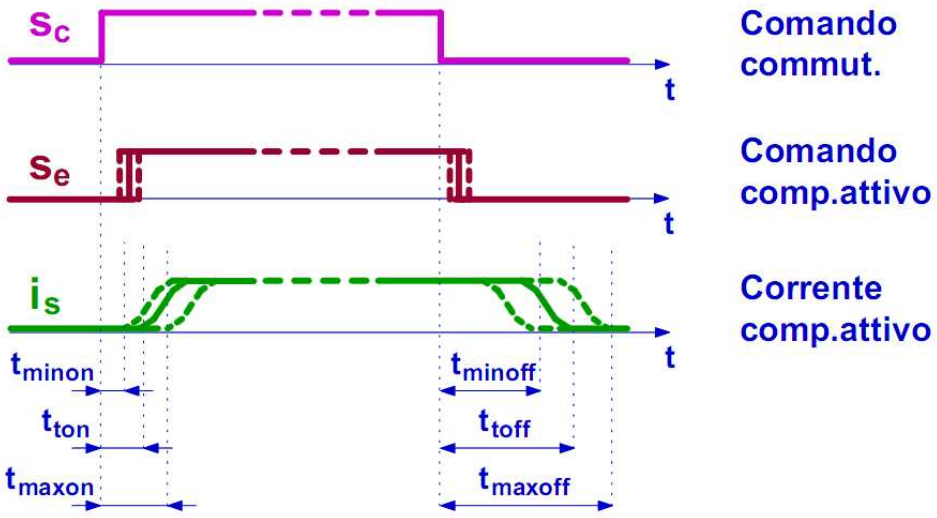

Fig. 1.5: Incertezze nei ritardi di commutazione (Fig. ricavata da [1])

Nel funzionamento del convertitore di tensione si deve evitare la condizione in cui siano conduttori contemporaneamente, anche per brevi istanti, entrambi i componenti attivi di commutazione (Fig. 1.6). In tali condizioni infatti si stabilisce una condizione di corto circuito tra le due alimentazioni $E/2$, e la corrente è limitata soltanto dalle impedenze Z_s dell'alimentazione e dal guadagno di corrente dei componenti attivi. La corrente quindi raggiunge valori assai elevati, che provocano dissipazioni pericolose nei componenti attivi.

Nelle commutazioni, è quindi necessario assicurarsi che T_1 non venga acceso prima che T_2 sia completamente spento, e viceversa per l'accensione di T_2. E' necessario quindi dare

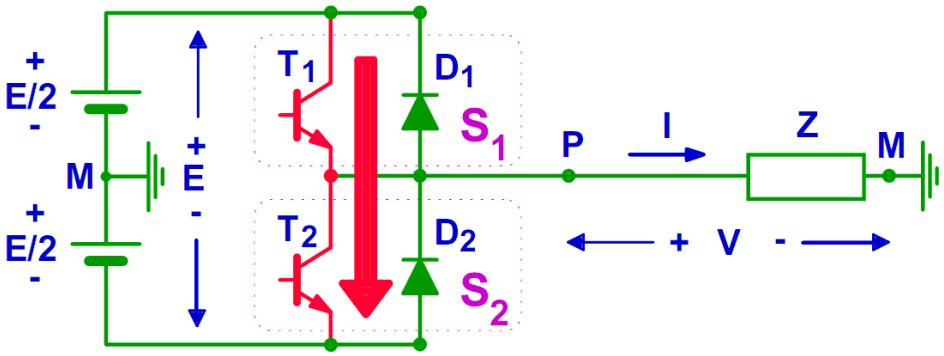

Fig. 1.6: Corto circuito di alimentazione (Fig. ricavata da [1])

i comandi di commutazione tenendo conto dell'esistenza dei ritardi t_{ton}, t_{toff}, descritti in precedenza, tra i segnali di comando e l'effettiva commutazione degli interruttori attivi. Il problema è complicato dal fatto che tali ritardi sono noti con una certa incertezza, per cui possono variare l'uno tra t_{minon} e t_{maxon} e l'altro tra t_{minoff} e t_{maxoff}. Per evitare la conduzione contemporanea, i comandi di accensione S_{c1}, S_{c2} degli interruttori S_1, S_2 vanno dati con un certo tempo di ritardo t_{dead} (detto usualmente 'tempo morto', in inglese 'dead time') rispetto al segnale di comando S_{CC} generato dal sistema di modulazione e quindi dopo che si è dato il comando di spegnimento all'altro interruttore. Come si vede in Fig. 1.7, l'introduzione dei tempi morti accorcia di t_{dead} il tempo di comando dello stato di accensione per entrambi gli interruttori. Perciò tale tempo $\delta_{c1}T_s$ per S_{c1} è inferiore a δT_s e $\delta_{c2}T_s$ per S_{c2} è inferiore a $(1-\delta)T_s$.

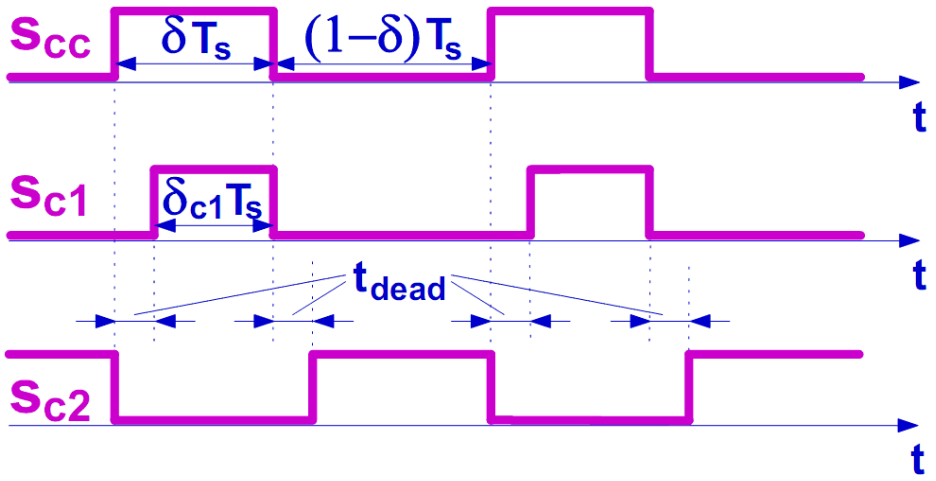

Fig. 1.7: Commutazioni: tempi morti (Fig. ricavata da [1])

A rigore, il minimo tempo morto che assicura la non contemporanea conduzione è:

$$|t_{dead}| = |t_{maxoff} - t_{minon}| \tag{1.3}$$

In pratica, si adottano tempi morti sensibilmente superiori (almeno $t_{dead} > t_{maxoff}$), per avere ampi margini di affidabilità. Introducendo i tempi morti nella generazione dei segnali di comando S_{c1} e S_{c2}, si evita la conduzione contemporanea ma si introduce ad ogni commutazione un intervallo in cui entrambi i componenti attivi T_1 e T_2 sono interdetti. Gli effetti di tale condizione si possono valutare esaminando in dettaglio lo svolgimento del

processo di commutazione. A questo fine, con riferimento alla Fig. 1.8, si supponga che T_1 stia conducendo la corrente I (positiva, uscente verso il carico) e che venga comandata la commutazione, spegnendo l'interruttore T_1, accendendo S_2 e trasferendo la corrente I da S_1 a S_2. A causa della commutazione la tensione di uscita $v(t)$, che inizialmente valeva $+E/2$, cambia segno e diventa pari a $-E/2$. Poichè il carico alla frequenza di commutazione ha caratteristiche induttive, la corrente che lo percorre non potrà avere cambiamenti istantanei. Essa dunque conserva il suo segno anche dopo lo spegnimento di S_1 e percorre S_2 con un verso tale da far condurre il diodo D_2 , mentre T_2 non conduce corrente, qualunque sia il comando applicato al suo elettrodo di controllo. In particolare, il diodo D_2 assicura una via di percorrenza alla corrente di carico immediatamente dopo che S_1 ha cessato di condurre e T_2 è ancora interdetto a causa dell'introduzione dei tempi morti.

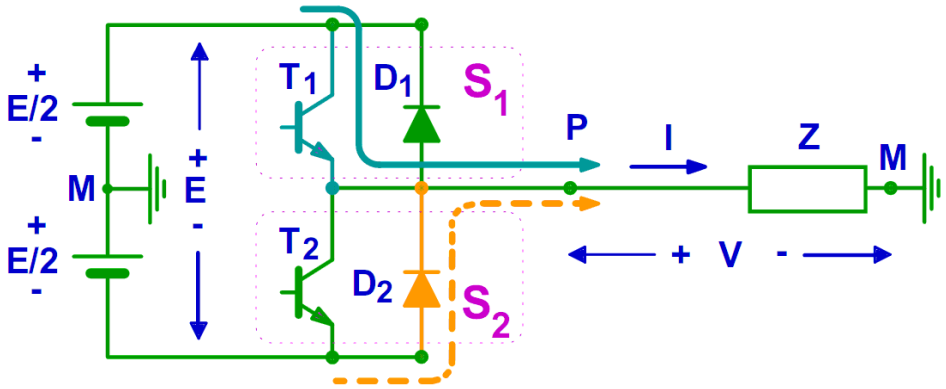

Fig. 1.8: Corrente I positiva (uscente) Commutazione tra T_1 e D_2 (Fig. ricavata da [1])

Se, come succede nella maggioranza dei casi, la corrente rimane dello stesso segno e non si annulla per tutta il tempo in cui S_1 è spento, la conduzione del diodo D_2 continua fino a quando S_1 viene riacceso. Quando S_1 torna a condurre (e supponendo che T_2 sia interdetto), si verifica la commutazione inversa ed il diodo D_2 si interdice. La tensione di uscita torna al valore $+E/2$. E' importante rilevare che in tutto questo processo il comando di T_2 non ha alcuna influenza, purché esso passi all'interdizione prima della riaccensione di S_1 e quindi di T_1. Un comportamento analogo, ma con funzioni invertite tra i due interruttori, si ha per corrente di carico negativa. Gli andamenti dei segnali di comando S_{c1}, S_{c2}, delle correnti negli interruttori i_{s1}, i_{s2} e della tensione di uscita $v(t)$ descritti fin qui per corrente I positiva sono mostrati in Fig. 1.9. In tale figura, per semplicità, si è supposto che, durante tutto il processo di commutazione, la corrente di carico rimanga costante al valore I (il che equivale a supporre che l'ondulazione triangolare di corrente abbia ampiezza trascurabile). Si è supposto ancora che le commutazioni nei componenti attivi e nei diodi avvengano con tempi trascurabili, con transizioni istantanee di tensione e di corrente da uno stato all'altro.

Dalla Fig. 1.9 si può constatare che, come già detto, per effetto dei tempi morti, la durata $\delta c_1 T_s$ del comando S_{c1} di accensione di S_1 è

$$\delta_{c1} T_s = \delta T_s - t_{dead} \tag{1.4}$$

Inoltre, per effetto dei ritardi di trasmissione e di quelli del componente attivo, la durata $\delta s_1 T_s$ della conduzione di T_1 risulta

$$\delta_{s1} T_s = \delta_{c1} T_s - t_{on} + t_{off} \tag{1.5}$$

che corrisponde normalmente ad un incremento di δ_{s1} rispetto a δ_{c1}.

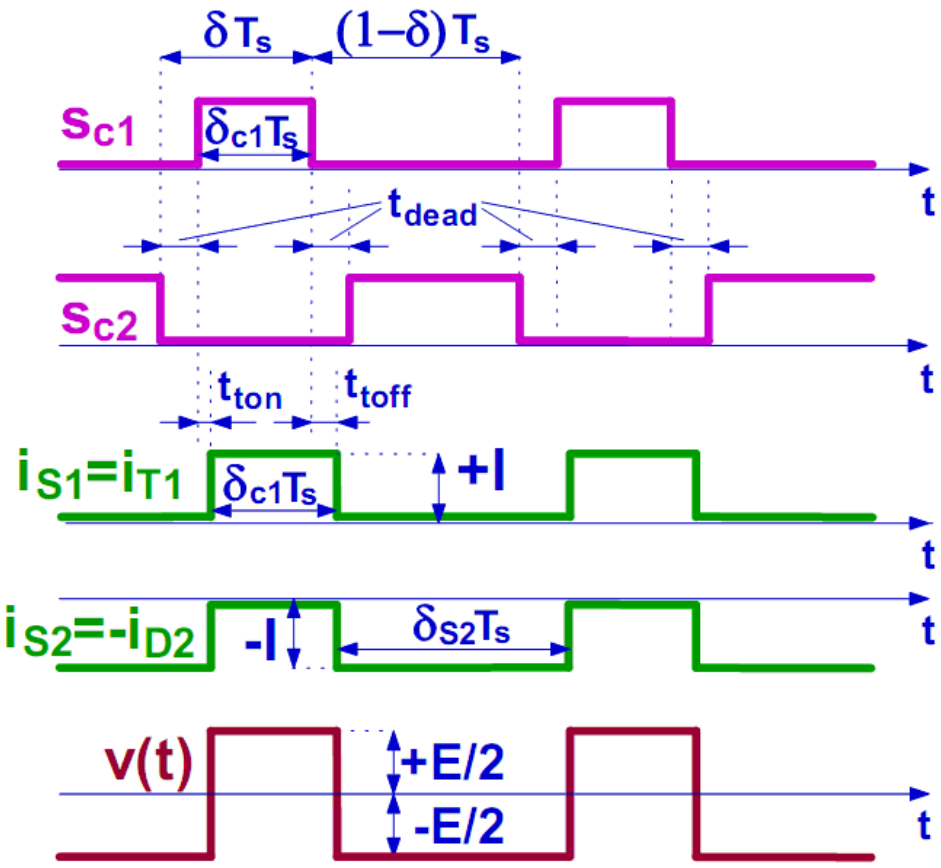

Fig. 1.9: Commutazioni (Fig. ricavata da [1])

La durata dell'impulso positivo della tensione di uscita $v(t)$ è ancora $\delta_{s1}T_s$, e quindi δ_{s1} è il duty-cycle effettivo che si ottiene complessivamente per effetto dei tempi morti e dei ritardi di trasmissione e di commutazione del componente attivo. La durata di conduzione dell'interruttore S_2 vale $(1-\delta_{s1})Ts$ ed è di fatto indipendente dal comando dato al componente attivo T_2. Si ricavano, pertanto:

$$\delta_{s1}T_s = \delta T_s - (t_{dead} + t_{on} - t_{off}) \tag{1.6}$$

$$\delta_{s2}T_s = (1-\delta)T_s + (t_{dead} + t_{on} - t_{off}) \tag{1.7}$$

Indicando con $t_{deadt} = (t_{dead} + t_{on} - t_{off})$ il tempo morto totale, dalle relazioni precedenti si ottengono:

$$\delta_{s1} = \delta - \frac{t_{dead}}{T_s} \tag{1.8}$$

$$\delta_{s2} = (1-\delta) + \frac{t_{dead}}{T_s} \tag{1.9}$$

Poiché di solito si prende t_{dead} alquanto maggiore di t_{on} e di t_{toff}, t_{dead} è normalmente positivo e quindi l'effettivo duty cycle δ_{s1} è minore di quello comandato δ. Ne consegue una diminuzione della tensione media generata nel periodo di modulazione. Infatti ponendo nella

$$v_{med} = (\delta_{s1} - \frac{1}{2})E \tag{1.10}$$

il valore effettivo δ_{s1}, si ottiene

$$v_{med} = (\delta - \frac{1}{2})E - \frac{t_{dead}}{T_s}E \tag{1.11}$$

Quindi esiste una diminuzione del valore medio effettivo di tensione, rispetto al valore medio teorico, pari a $\frac{T_{dead}}{T_s}E$.

Se la corrente I è negativa, si ha un comportamento analogo ma con funzioni invertite dei due interruttori. La durata di conduzione di S_2 viene ridotta e quella di S_1 viene aumentata. Le 1.6 e 1.7 diventano

$$\delta_{s2}T_s = (1-\delta)T_s - (t_{dead} + t_{on} - t_{off}) \tag{1.12}$$

$$\delta_{s1}T_s = \delta T_s + (t_{dead} + t_{on} - t_{off}) \tag{1.13}$$

Di conseguenza si ottengono:

$$\delta_{s2} = (1-\delta) - \frac{t_{dead}}{T_s} \tag{1.14}$$

$$\delta_{s1} = \delta + \frac{t_{dead}}{T_s} \tag{1.15}$$

Infine si ottiene il seguente valore medio di tensione sul carico:

$$v_{med} = (\delta_{s1} - \frac{1}{2})E \tag{1.16}$$

$$v_{med} = (\delta - \frac{1}{2})E + \frac{t_{dead}}{T_s}E \tag{1.17}$$

In questo caso, con corrente negativa, si ha dunque un aumento della tensione media nel periodo di modulazione. Perciò, quando la corrente cambia segno, a parità di duty-cycle assegnato, si ha una brusca variazione del duty-cycle effettivo e del valore della tensione di uscita dell'inverter. Tale effetto è causa di errori e deformazioni nell'andamento della tensione di uscita del convertitore, con conseguenze non sempre trascurabili. In Fig. 1.10 sono mostrati gli effetti dei tempi morti sulla tensione di uscita dell'inverter. Come detto precedentemente la presenza dei tempi morti produce un'alterazione del valore medio della tensione prodotta dall'inverter, il cui verso dipende da quello della corrente e la cui ampiezza è (teoricamente) indipendente da quella dalla corrente stessa e dal valore del duty cycle. Infatti, per correnti positive abbiamo:

$$\Delta v_{med} = v_{med} - v_{medt} = -\frac{t_{dead}}{T_s}E \tag{1.18}$$

Mentre per correnti negative abbiamo:

$$\Delta v_{med} = v_{med} - v_{medt} = +\frac{t_{dead}}{T_s}E \tag{1.19}$$

Come mostrano queste relazioni, l'entità delle variazioni di tensione media dipende soltanto dal rapporto tra il tempo morto t_{deadt} ed il periodo di modulazione T_s.

In Fig. 1.10, v_{med} rappresenta l'andamento della tensione di uscita dell'inverter, mediato in ogni periodo di modulazione, mentre v_{medt} è quello teorico che si avrebbe in assenza di tempi morti. Notiamo i gradini che si verificano in v_{med} in concomitanza ai passaggi per lo zero della corrente, a cui corrispondono cambi di segno di Δv_{med}. Tali distorsioni, come anticipato precedentemente, dipendono esclusivamente dalla formula $\pm\frac{t_{deadt}}{T_s}$ Quindi

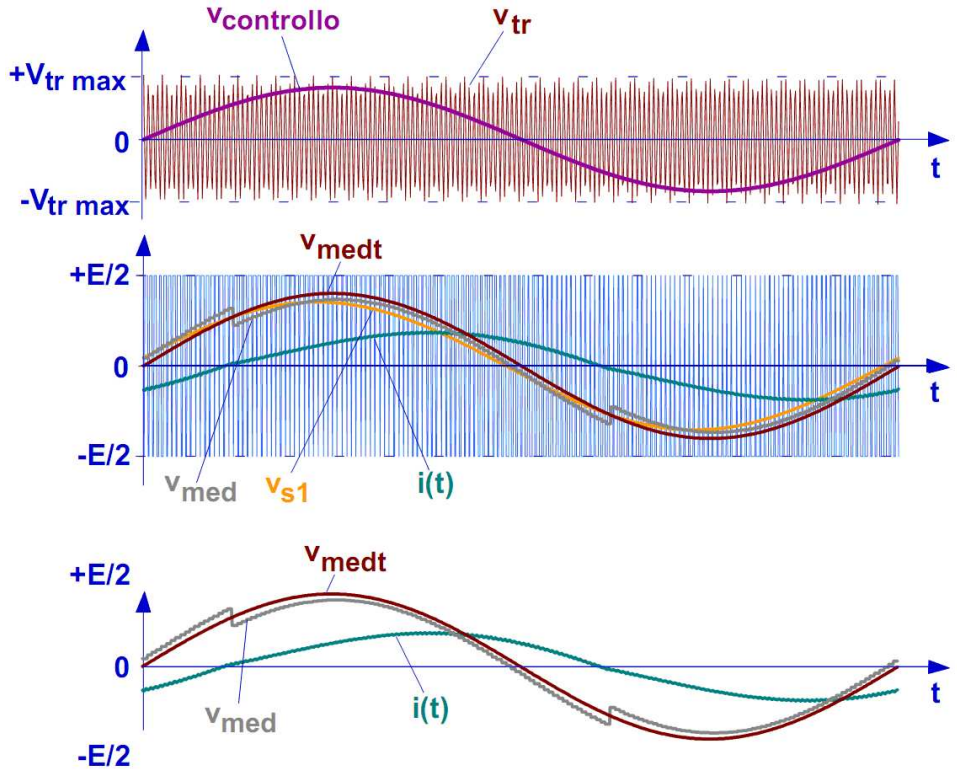

Fig. 1.10: Effetto dei tempi morti sulla tensione media (Fig. ricavata da [1])

si può ottenere una sensibile riduzione di tali effetti diminuendo la frequenza di modulazione o diminuendo i tempi morti. In Fig. 1.10 si è anche riportato l'andamento della prima armonica v_{s1} della tensione di uscita. Come si vede, a causa della deformazione causata dai tempi morti, essa non risulta in fase con la modulante sinusoidale $v_{controllo}$. La distorsione di tensione causa una distorsione di corrente, e questo si traduce successivamente in pulsazioni di coppia sull'albero motore. In alcune applicazioni industriali è stato riportato che, in alcune determinate condizioni, ciò può comportare problemi di instabilità tra l'azionamento ed il motore. Diversamente dal segnale modulato, che è impostato dall'algoritmo di controllo per ottenere una determinata ampiezza di tensione sul carico, l'ampiezza della distorsione di tensione non è influenzata dal controllo, rimanendo costante in ogni condizione operativa. Ciò significa che, dal punto di vista del rapporto segnale/rumore, il problema della distorsione è più sentito e grave quando la tensione di riferimento imposta dall'algoritmo di controllo è piccola. Nei motori in alternata ciò può coincidere con situazioni in cui la velocità del rotore è molto bassa, tali che il momento d'inerzia del rotore stesso non riesce a filtrare le pulsazioni di coppia, rendendole ben visibili sul carico meccanico.

1.2 Motivazione della misura di tensione al posto della stima tramite compensazione delle non linearità dell'inverter

1.2.1 Problematiche negli azionamenti e introduzione alle tecniche di compensazione dei tempi morti

Dallo studio delle tecniche di modulazione e delle varie tipologie di convertitori è emerso chiaramente come sia possibile applicare una tensione di ampiezza e frequenza variabile

al motore, a partire dai riferimenti di tensione normalmente generati all'interno degli algoritmi di controllo. Come visto nella sottosezione 1.1.3, un inverter reale è composto da interruttori caratterizzati da cadute di tensione in conduzione, che richiedono del tempo per chiudersi ed aprirsi. Si prenda come esempio un IGBT; esso durante la fase di chiusura presenta una tensione non nulla ai suoi capi, che costituisce di fatto una tensione persa non trasferita al carico. Inoltre, la sua capacità di aprirsi velocemente è di molto ridotta a causa del fenomeno della coda di corrente, che costringe ad aspettare anche alcuni microsecondi prima che l'IGBT possa considerarsi veramente aperto. Inoltre, è bene anche ricordare che il bus in continua non è un elemento a rigidità infinita, ossia c'è da aspettarsi che la tensione ai suoi capi possa fluttuare leggermente, senza alcuna regola prestabilita. Gli algoritmi di controllo moderni, pertanto, devono prevedere obbligatoriamente alcune contromisure a questi problemi reali. Ad esempio, la misura della tensione di bus Udc è effettuata in tempo reale ed utilizzata nel calcolo dei tempi di attivazione delle fasi di un convertitore in modo da compensare eventuali fluttuazioni di tensione. Un'altra fondamentale contromisura consiste (come visto nella sottosezione 1.1.3) nel ritardare l'accensione degli IGBT in un inverter trifase, in modo da dare all'IGBT complementare sullo stesso ramo, il tempo necessario alla sua completa apertura, evitando cortocircuiti della tensione di bus. Così facendo viene introdotto un tempo morto (dead time). Si è osservato nella sottosezione 1.1.3 come l'introduzione di un tempo morto faccia deviare la tensione reale sul carico rispetto a quella di riferimento desiderata. Si può intuire come i problemi maggiori si hanno per tensioni di riferimento molto piccole, quando la distorsione di tensione causata dai tempi morti è percentualmente più rilevante rispetto al caso di tensioni elevate. Si è inoltre fatto notare come la distorsione aumenti con l'aumento della frequenza di commutazione del convertitore, introducendo armoniche di tensione che, non opportunamente compensate, potrebbero causare problemi di stabilità del sistema. Oltre agli eventuali problemi di stabilità, la conoscenza della distorsione legata ai tempi morti, e la sua corretta compensazione, è importante in tutti gli algoritmi sensorless. Questi algoritmi non necessitano della misura della coppia all'albero motore, la quale viene stimata attraverso modelli matematici del motore controllato. Il corretto funzionamento di tali modelli si basa sulla conoscenza della tensione applicata ai capi del motore; se essa è diversa dalla tensione di riferimento richiesta dagli algoritmi di controllo, la stima della coppia ne verrà influenzata, specialmente a basse velocità. In quanto si è visto che a basse velocità (e quindi basse tensioni) le non linearità risultanto più penalizzanti. Il problema della distorsione legata ai tempi morti può essere comunque superato mediante un accurato studio teorico ed un'adeguata strategia di compensazione, che permetta di considerare la tensione di riferimento generata dagli algoritmi di controllo come la tensione effettivamente generata sul carico. Ciò permette il quasi completo ripristino della linearità del sistema di controllo con benefici per la stabilità dello stesso. Nei paragrafi successivi saranno descritte alcune tecniche di compensazione delle non linearità. Dopo una breve descrizione delle varie tecniche, verranno elencati i difetti di ciascuna e si giungerà alla conclusione che per ottenere le massime prestazioni dall'algoritmo sensorless, si preferisce misurare direttamente le tensioni applicate al motore (e non stimarle).

1.2.2 Strategia di compensazione classica

La più semplice strategia adottabile, largamente utilizzata negli azionamenti industriali, prevede il semplice aggiustamento delle tensioni di riferimento tendendo conto dell'effetto dei tempi morti. In altre parole, una volta ottenuti i riferimenti medi di tensione u^*_{san}, u^*_{sbn} e u^*_{scn} dall'algoritmo di controllo e prima di procedere alla traduzione in impulsi per gli IGBT dell'inverter di tensione, l'algoritmo di controllo stesso controbilancia l'effetto teorico dei tempi morti, sommando ai riferimenti di tensione le quantità 1.20.

$$U_{ancomp} = u^*_{san} - u_{san} = \frac{t_d}{T_s} U_{dc} sgn(i_{sa})$$

$$U_{bncomp} = u^*_{sbn} - u_{sbn} = \frac{t_d}{T_s} U_{dc} sgn(i_{sb}) \tag{1.20}$$

$$U_{cncomp} = u^*_{scn} - u_{scn} = \frac{t_d}{T_s} U_{dc} sgn(i_{sc})$$

dove:

- U_{xncomp} è la tensione di compensazione risultante per la fase x
- $u^*_{sxn} = \frac{T_{x,ON}}{T_s} U_{dc}$ è la tensione di riferimento media della fase x
- u_{sxn} è la tensione media della fase x effettiva (considerando la variazione di tensione di fase introdotta dai tempi morti)
- $sgn(i_{sx}) = 1$ se $i_{sx} > 0$

  $sgn(i_{sx}) = -1$ se $i_{sx} < 0$

  con i_{sx} = corrente della fase x

Questa soluzione è senz'altro la più immediata, anche se ha molti svantaggi. In primo luogo non descrive completamente il fenomeno dei tempi morti, trascurando gli effetti parassiti che possono incidere notevolmente specie per piccole tensioni sul carico. Inoltre, le 1.20 considerano un tempo t_d sempre costante: nella realtà, invece, esso è dipendente dalle condizioni di carico, nel senso che i fronti di salita e discesa della tensione dipendono dal punto di lavoro degli IGBT. Infine, tale tecnica di compensazione, produce una errata compensazione qualora la corrente cambi segno all'interno del periodo di commutazione.

1.2.3 Compensazione basata sull'utilizzo di una FPGA

La strategia di compensazione classica spiegata nella sottosezione 1.2.2 produce una errata compensazione qualora la corrente cambi segno all'interno del periodo di commutazione. Al fine di considerare gli eventuali cambiamenti di segno della corrente all'interno del periodo della PWM, è necessario sovracampionare la corrente stessa; a questo proposito, una delle soluzioni adottabili prevede l'utilizzo di una Field Programmable Gate Array (FPGA), caratterizzate da velocità di acquisizione dei dati e di calcolo di gran lunga superiori ad un normale microprocessore. La strategia adottabile con l'utilizzo di una FPGA prevede il campionamento della corrente di fase pochi istanti prima che l'interruttore della fase stessa cambi il suo stato (da ON a OFF o viceversa). Se la corrente risulta positiva, allora il fronte negativo del comando all'interruttore inferiore deve essere anticipato di t_d (si osservino la Fig. 1.11 e la Fig. 1.12). Se la corrente risulta negativa, invece, è il fronte negativo del comando all'interruttore superiore che deve essere anticipato di t_d (si vedano la Fig. 1.11 e la Fig. 1.13). Alla luce di ciò, l'acquisizione della corrente da parte dell'FPGA deve avvenire alcuni nanosecondi prima del possibile fronte anticipato (istanti $t_{x,1}$ e $t_{x,2}$ in Fig. 1.12 e in Fig. 1.13). Si osservi, pertanto, che la corrente viene campionata praticamente in tempo reale due volte per ogni periodo, assicurando una corretta

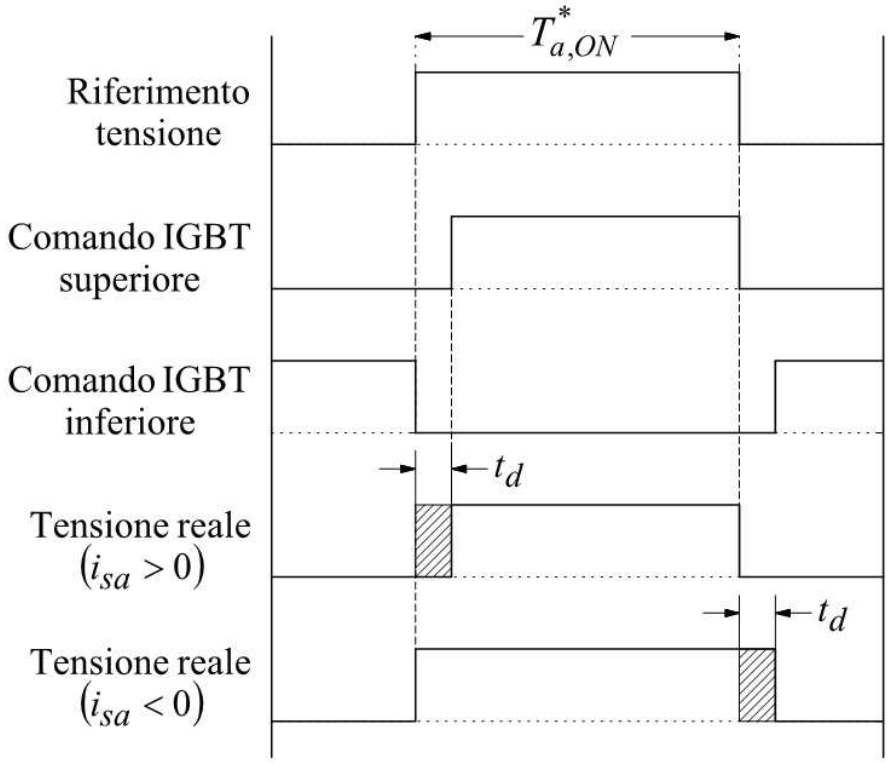

Fig. 1.11: Introduzione dei tempi morti nelle commutazioni dell'inverter di tensione (Fig. ricavata da [2])

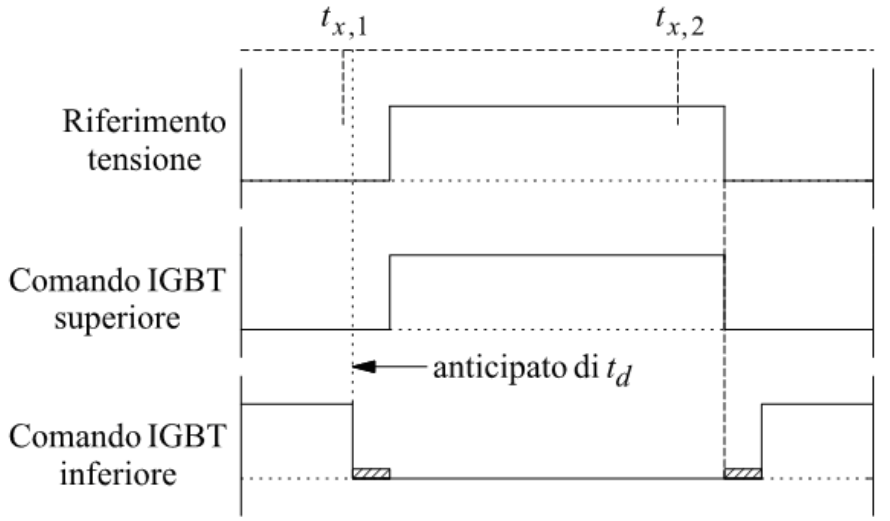

Fig. 1.12: Strategia di compensazione dei tempi morti (con corrente positiva) (Fig. ricavata da [2])

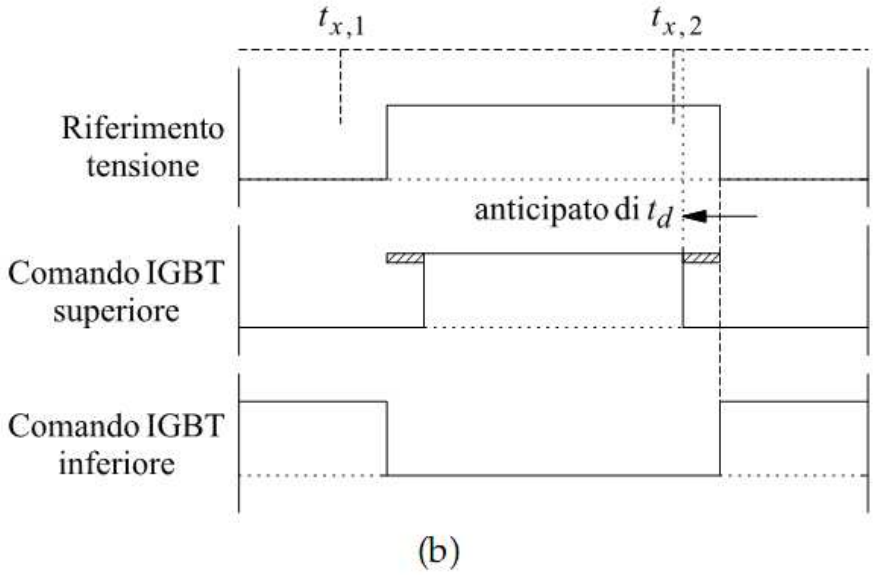

Fig. 1.13: Strategia di compensazione dei tempi morti (con corrente negativa) (Fig. ricavata da [2])

compensazione anche in caso di cambi di segno della corrente all'interno del periodo di campionamento.

Alcuni test sperimentali hanno dimostrato la validità della procedura appena qui presentata rispetto ad una compensazione classica. Nonostante ciò, questo genere di compensazione non tiene in considerazione le problematiche relative agli effetti parassiti degli IGBT, che sono causa di armoniche spurie nello spettro anche a compensazione avvenuta.

1.2.4 Effetti parassiti degli IGBT negli inverter di tensione

La descrizione matematica della distorsione legata ai tempi morti descritta nella sottosezione 1.1.3 ipotizza che gli interruttori costituenti l'inverter trifase (siano essi IGBT o MOSFET) abbiano un comportamento ideale. Purtroppo, specialmente nel caso degli IGBT, ciò non corrisponde alla realtà. Esistono infatti degli effetti parassiti, come ad esempio le capacità non lineari tra i terminali degli IGBT, che rendono il fenomeno della commutazione fortemente non lineare e difficilmente descrivibile attraverso un modello matematico, specialmente quanto l'ampiezza della corrente in gioco è molto piccola. Il problema di fondo è che gli effetti parassiti modificano e in qualche caso invalidano il modello matematico che descrive la distorsione legata ai tempi morti, specialmente durante l'attraversamento dello zero di una delle tre correnti di fase. Un esempio tipico è legato proprio alle capacità parassite negli IGBT. Una corrente positiva/negativa di piccola ampiezza che scorre al/dal motore non è in grado di agganciare completamente la tensione sul motore al potenziale negativo/positivo del bus in continua, a causa della carica residua presente nelle capacità. La tensione, invece che passare istantaneamente al valore negativo/positivo di tensione, rimane a dei valori intermedi. La situazione è rappresentata in Fig. 1.14

Una delle conseguenze più immediate degli effetti parassiti negli IGBT è che le classiche compensazioni dei tempi morti, che si basano sulla compensazione in feedforward della tensione (vedi sottosezione 1.2.2) risultano non efficaci per la compensazione in tempo reale quando le correnti sono molto basse. Senza considerare che, ad esempio, la compensazione classica basata sulle 1.20 necessita della conoscenza perfetta dell'attraversamento della

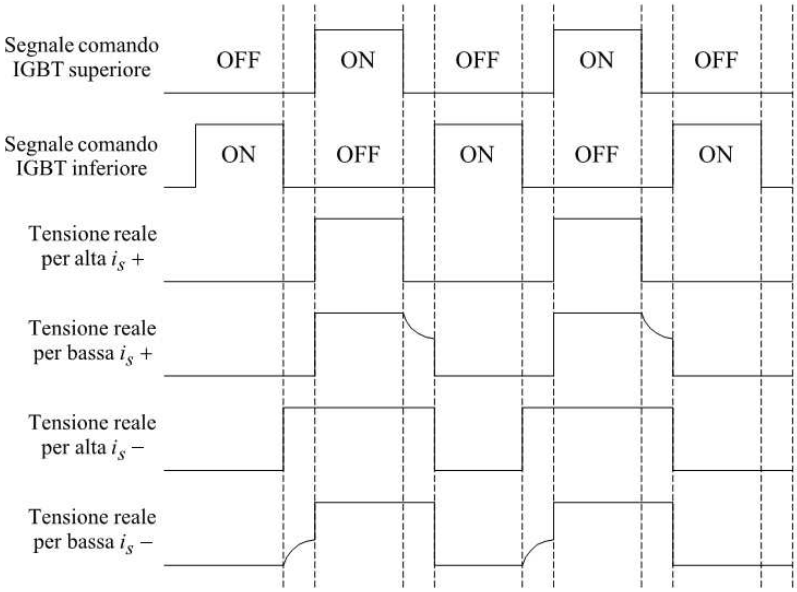

Fig. 1.14: Effetti parassiti degli IGBT sulla tensione di fase generata dall'inverter (Fig. ricavata da [2])

corrente di fase, che è piuttosto difficile da ottenere a causa sia della distorsione di corrente provocata dagli effetti parassiti che del rumore sovrapposto alle misure. Nella sottosezione 1.2.5 verrà illustrato come la compensazione degli effetti parassiti è possibile attraverso alcune procedure particolari da eseguire sul motore stesso.

1.2.5 Compensazione dei tempi morti con tecniche repetitive

Le strategie di compensazione finora presentate non hanno mai tenuto in considerazione il fatto che, per quanto detto nella sottosezione 1.2.4, una compensazione basata sulla descrizione matematica in 1.20 non permette un'adeguata compensazione dei tempi morti quando la corrente attraversa lo zero. D'altra parte, è molto difficile valutare in termini matematici tutti i fenomeni che incorrono quando la corrente assume valori molto ridotti. Pertanto, una soluzione plausibile potrebbe essere quella di osservare il fenomeno nell'azionamento stesso, registrarne in qualche modo l'andamento e compensarlo in tempo reale nel normale funzionamento del motore. Le procedure che leggono autonomamente, senza alcun aiuto da parte dell'operatore esterno, il comportamento dell'azionamento e ne ricavano un risultato attraverso una serie di calcoli matematici (compensazione dei tempi morti, ma anche stime dei parametri del motore e non solo) sono dette procedure di self-commissioning. Al giorno d'oggi gli azionamenti industriali sono dotati di procedure di questo tipo in grado di tarare autonomamente i regolatori di corrente e velocità dell'azionamento, talvolta senza alcun intervento manuale. E' possibile dotare un azionamento di una procedura di self-commissioning per una compensazione corretta dei tempi morti anche durante l'attraversamento dello zero delle correnti. La base di partenza è la considerazione che la descrizione matematica in 1.20 può essere trasformata in un sistema di riferimento rotante, e più precisamente in un sistema di riferimento solidale con l'angolo della corrente, che qui chiameremo θ_i, tale che la terna di correnti trifase nel motore sia:

$$i_{sa} = Icos(\theta_i)$$

$$i_{sb} = Icos(\theta_i - \frac{2\pi}{3}) \tag{1.21}$$

$$i_{sc} = Icos(\theta_i - \frac{4\pi}{3})$$

dove I è l'ampiezza della corrente. Trasformando le 1.20 in tale sistema di riferimento dq, si ottengono le seguenti espressioni per la tensione di distorsione legata ai tempi morti:

$$u_{comp,d}(\theta_i) = \frac{4t_d}{\pi T_c} U_{dc} \left[1 - \sum_{n=1}^{+\infty} (-1)^n \frac{2}{36n^2 - 1} cos(6n\theta_i) \right]$$

(1.22)

$$u_{comp,q}(\theta_i) = \frac{4t_d}{\pi T_c} U_{dc} \left[\sum_{n=1}^{+\infty} (-1)^n \frac{12n}{36n^2 - 1} sin(6n\theta_i) \right]$$

La dimostrazione delle 1.22 è riportata in [2]. Le espressioni 1.22 mostrano che, a parte il termine costante in $u_{comp,d}$, la distorsione legata ai tempi morti può essere espressa come una somma di seste armoniche e suoi multipli. Questo andamento armonico si somma al comportamento non lineare degli IGBT che insorge quando una corrente di fase attraversa lo zero, e che non è adeguatamente descritto in forma matematica. Tuttavia, essendoci tre fasi nell'inverter di tensione e dunque sei attraversamenti dello zero della corrente in un periodo completo dell'angolo θ_i, il comportamento non lineare degli IGBT è sicuramente descritto da una sesta armonica (e suoi multipli), esattamente come le componenti descritte in 1.22. Questa fondamentale caratteristica viene utilizzata per sviluppare la procedura di self-commissioning per la compensazione dei tempi morti. La procedura di self-commissioning si basa pesantemente sull'utilizzo di un controllo noto con il nome di repetitive. In questo capitolo non ci si pone l'obbiettivo di spiegare dettagliatamente tale tecnica. Si concentrerà invece l'attenzione, sui miglioramenti apportati rispetto alle tecniche spiegate nelle sottosezioni 1.2.2 e 1.2.3, e si osserveranno le problematiche che affliggono tale soluzione. Per approfondire tale metodo di compensazione si rimanda alla [2]. La compensazione con tale tecnica, si basa sulla misurazione delle correnti di statore, tali correnti vengono utilizzate nell'algoitmo repetitive, per generare la tensione di compensazione, che verrà sommata alla tensione di riferimento. La somma risultante crea il riferimento delle tensioni di fase compensato in maniera corretta, tenendo conto sia dei tempi morti, che degli effetti parassiti degli interruttori di potenza. La procedura di self-commissioning è eseguita fino a che l'input dei blocchi repetitive, cioè le componenti di sesta armonica, sono ridotte al di sotto di uno specifico intervallo, impostato come percentuale del valore nominale della corrente nel motore. Mentre la procedura è in esecuzione, la tensione di compensazione è salvata in tre buffer circolari, uno per ognuna delle tre fasi. Una volta raggiunta la convergenza, i tre buffer circolari sono trasferiti in tre look-up table (LUT), una per ogni fase del motore, per essere utilizzate nel normale funzionamento dell'azionamento. I valori salvati nelle LUT saranno richiamati sulla base della corrente di fase misurata, e costituiranno la tensione di compensazione dei tempi morti per ognuna delle tre fasi. Per correggere eventuali variazioni della tensione di bus, la tensione in output viene corretta con il rapporto tra la tensione di bus misurata e la tensione di bus presente al momento del salvataggio delle LUT.

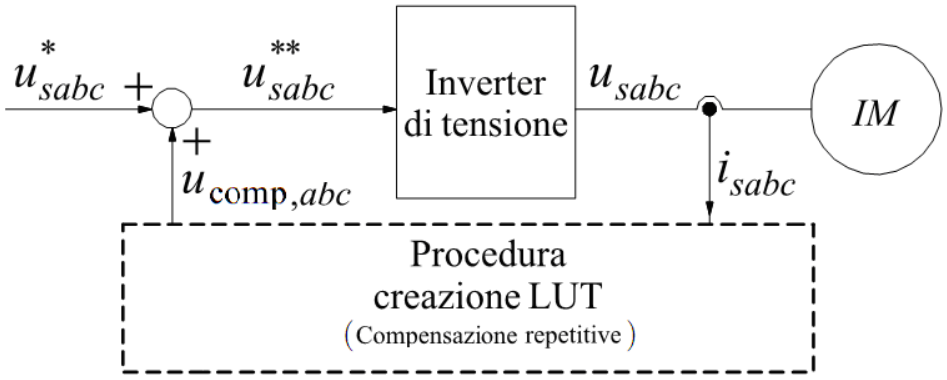

Fig. 1.15: Schema della procedura self-commissioning per la compensazione dei tempi morti basata sul controllo Repetitive (Fig. ricavata da [2])

La peculiarità delle LUT generate è una compensazione più morbida nell'intorno dello zero della corrente di fase, al contrario della descrizione matematica fornita in 1.20 che compenserebbe con un gradino di tensione nello zero. Ciò è legato al fatto che questa compensazione tiene adeguatamente in considerazione i comportamenti non lineari degli IGBT nell'intorno dello zero. Di conseguenza, la compensazione dei tempi morti nell'intorno dello zero della corrente di fase risulta meno critica, anche perché un piccolo errore nell'acquisizione della corrente provocherebbe solamente una piccola variazione della tensione di compensazione. Questo comportamento è differente rispetto alla classica compensazione a gradino, che nell'intorno dello zero può far insorgere fenomeni di chattering e causare addirittura instabilità a base velocità se l'attraversamento dello zero non è adeguatamente identificato. E' importante sottolineare che, siccome le LUT sono generate per una specifica velocità, si accetta implicitamente che tale compensazione possa essere estesa anche ad altre velocità, trascurando un eventuale contributo dipendente dalla velocità stessa.

1.2.6 Compensazione dei tempi morti con misure a motore fermo

Lo svantaggio principale del metodo di compensazione descritto nella sottosezione 1.2.5 è che esso richiede la rotazione del motore, e ciò non è sempre possibile nella realtà. Esistono molti casi in cui l'azionamento elettrico viene installato in sistemi industriali nei quali il motore è già collegato al carico meccanico, e non più scollegabile pena la sospensione della produzione. Tali situazioni si presentano tipicamente quando si cerca di aumentare le prestazioni e/o l'efficienza di vecchi motori già installati in linee di produzione, semplicemente cambiando la parte relativa all'elettronica di potenza e di controllo. E' dunque chiaro che, dovendosi presentare una tale situazione, una procedura di compensazione dei tempi morti che richieda la rotazione del motore risulta inadeguata agli scopi prefissati. Sarebbe dunque auspicabile che gli algoritmi di self-commissioning fossero in grado, per quanto possibile, di stimare i tempi morti mediante procedure a rotore fermo. Esistono fortunatamente dei metodi in grado di stimare i tempi morti senza generare un campo magnetico rotante e dunque senza far ruotare il motore. Questi algoritmi sfruttano, in particolare, il fatto che alimentando un motore (di qualunque tipologia) con una tensione continua (o meglio, continua in media per la presenza di un inverter di tensione), dopo un transitorio legato ai parametri del motore stesso si ottiene una corrente anch'essa continua. Il rapporto tra la tensione e la corrente risultanti è ovviamente relativo alla resistenza degli avvolgimenti di statore della macchina, e qualunque deviazione dalla semplice legge di Ohm deve dunque essere imputata alla presenza dei tempi morti nell'inverter di tensione. Uno dei sistemi più semplici per applicare una tensione continua in media è quello

di impostare un riferimento continuo positivo di tensione sul ramo a, ed un riferimento continuo negativo di ampiezza dimezzata sui rami b e c dell'inverter. In questo modo, la corrente circolerà con verso positivo lungo la fase a del motore, e verrà richiusa nelle fasi b e c le quali porteranno, rispettivamente, metà della corrente misurata sulla fase a (a meno di asimmetrie interne nelle fasi del motore). Un grafico sperimentale di $u_{sa} = f(i_{sa})$ ottenuto con questa procedura è riportato in Fig. 1.16.

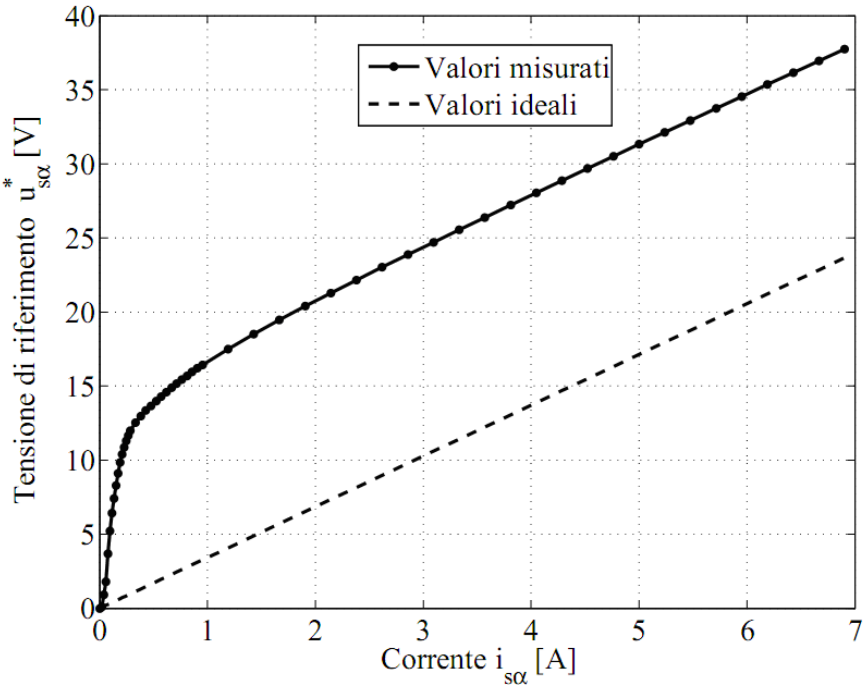

Fig. 1.16: Schema di compensazione on-line dei tempi morti mediante utilizzo di una look-up table (Fig. ricavata da [2])

Si osserva immediatamente che, a differenza della ideale legge di Ohm $u_{s\alpha} = R_S i_{s\alpha}$ riportata in tratteggio, con R_S resistenza di statore dell'avvolgimento a, la tensione di riferimento si discosta incurvandosi per basse correnti, e rimanendo parallela alla curva ideale per alte correnti. Ciò rispecchia essenzialmente quanto già detto nelle sottosezioni 1.1.3 e 1.2.2, e cioè che le tensioni di riferimento differiscono dalla tensione reale delle quantità riportate nelle 1.20. Particolarizzando il calcolo per $i_{sa} > 0$, $i_{sb} < 0$ ed $i_{sc} < 0$: ne risulta una differenza positiva, così come in Fig. 1.16. Se, tuttavia, le 1.20 fossero vere per ogni valore di corrente, la differenza tra la tensione di riferimento e la tensione reale sarebbe sempre costante, ossia le due curve sarebbero parallele anche per basse correnti. Tuttavia, come è stato discusso nella sottosezione 1.2.4, la descrizione matematica della distorsione legata ai tempi morti non è più corretta, a causa degli effetti parassiti dominanti, quando gli IGBT sono attraversati da basse correnti. La differenza tra la tensione di riferimento e la tensione reale è, di fatto, ciò che l'azionamento necessita come compensazione del fenomeno dei tempi morti. A partire dalle misurazioni di Fig. 1.16, focalizzando solamente la curva per alti valori di corrente, è possibile calcolare il valore della resistenza di statore mediante un semplice calcolo della pendenza della retta risultante. Una volta ottenuto il valore di R_S, è possibile tracciare la curva ideale (quella tratteggiata in Fig. 1.16) ed operare la differenza tra la curva misurata e quella ideale. La Fig. 1.17 riporta tale differenza. Siccome il comportamento dell'inverter risulta il medesimo anche per correnti negative, la differenza di Fig. 1.17 può essere estesa anche all'asse negativo, avendo l'accortezza di cambiare il segno in accordo alle 1.20. Il grafico completo è riportato in Fig. 1.18.

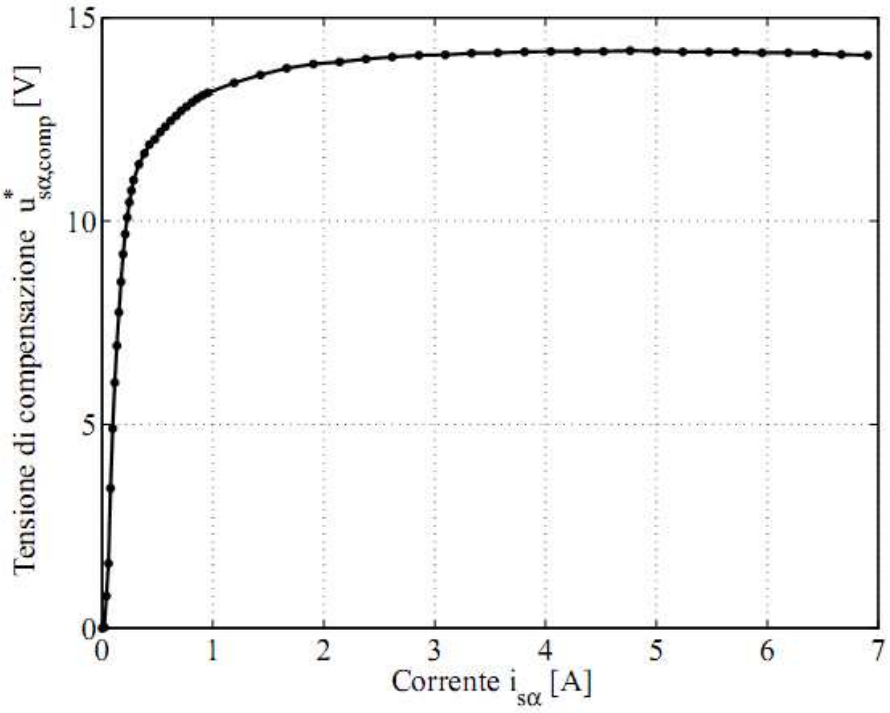

Fig. 1.17: Misure a motore fermo in funzione della corrente di fase (Fig. ricavata da [2]) (tensione di compensazione sul primo quadrante)

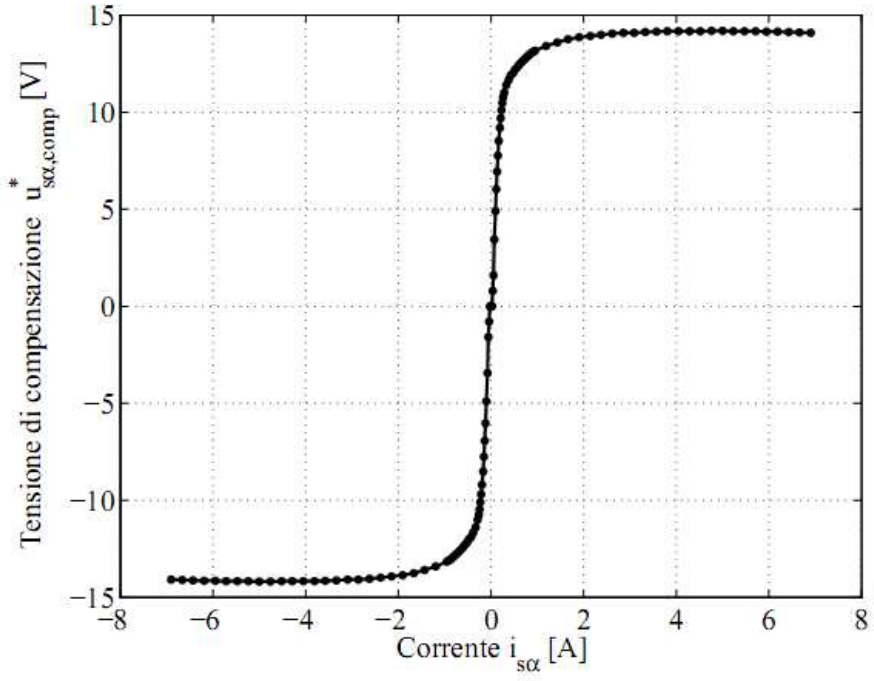

Fig. 1.18: Misure a motore fermo in funzione della corrente di fase (Fig. ricavata da [2]) (estensione ai quattro quadranti)

Il grafico ottenuto per l'intero range di correnti negative e positive può essere salvato in una tabella, alla quale l'algoritmo accede in tempo reale mediante la misura di corrente prelevata sul motore, per ottenere la compensazione di tensione necessaria ad annullare l'effetto dei tempi morti. Lo schema di principio è riportato in Fig. 1.19.

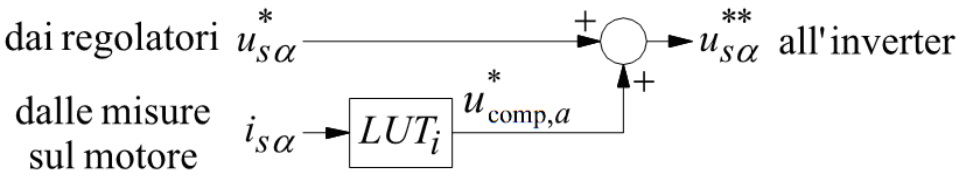

Fig. 1.19: Misure di tensione di riferimento a motore fermo in funzione della corrente di fase (Fig. ricavata da [2])

Occorre tener presente che la procedura descritta è legata ad un ben preciso valore della tensione di bus U_{dc}, e la tabella utilizzata in Fig. 1.19 farà riferimento al valore di U_{dc} (si osservino di nuovo le 1.20) presente al momento dell'attivazione della procedura di self-commissioning. Ciò significa che, qualora avvenissero delle variazioni della tensione di bus durante il normale funzionamento dell'azionamento, senza opportuni sistemi di correzione, la compensazione dei tempi morti non sarebbe più precisa, ciò comporterebbe l'iniezione di tensioni di riferimento non corrette. Qualora tali tensioni venissero utilizzate come parametri all'interno di algoritmi sensorless, l'errore di stima delle stesse, si rifletterebbe all'interno dell'algoritmo di stima della velocità all'albero motore.

1.2.7 Considerazioni e obiettivi del progetto

Nella breve descrizione delle varie modalità di stima delle tensioni tramite tecniche di compensazione dei tempi morti, abbiamo riportato per ciascuna tipologia delle problematiche che qui riassumiamo in breve:

1. non considerazione degli effetti parassiti degli interruttori elettronici di potenza
2. non considerazione delle possibili variazioni dei tempi morti (t_d)
3. errori di compensazione qualora la corrente cambi segno all'interno del periodo di commutazione
4. introduzione di problemi di chattering e instabilità alle basse velocità
5. obbligatoria rotazione del motore
6. errori causati da una possibile variazione della tensione di bus

Il fine ultimo del progetto è quello di misurare le tensioni sinusoidali reali fornite dall'inverter al motore, per poterle utilizzare, in una fase più avanzata, all'interno di un algoritmo sensorless per la stima di coppia del motore stesso. Per ottenere una stima accettabile si deve far sì che il progetto in tutte le sue fasi sia dotato di una notevole precisione, soprattutto nelle parti prive di eventuali controlli in retroazione capaci di annullare eventuali errori. Quindi, considerata la non sufficiente robustezza della stima delle tensioni tramite tecniche di compensazione delle non linearità dell'inverter, si è scelto di misurare direttamente le tensioni in uscita dall'inverter, attraverso un sistema di acquisizione dati, basato sull'utilizzo di una scheda FPGA.

Capitolo 2

HARDWARE

2.1 Schema generale dell'hardware utilizzato

Col fine di ottenere la misura delle tensioni concatenate fornite dall'inverter al motore, si è allestito un banco di prova composto dai seguenti elementi:

1. Computer per la gestione del software utilizzato sia per il controllo del motore, che per la misurazione delle tensioni concatenate fornite al motore stesso
2. Scheda dSPACE, per l'interfacciamento tra il computer e l'inverter di tensione
3. Bus in continua per la regolazione della tensione proveniente dalla rete
4. Inverter trifase PWM che fornisce le tre tensioni di fase al motore
5. Scheda analogica, della quale è stato esclusivamente utilizzato lo stadio di attenuazione delle tensioni di ingresso.
6. Scheda di interfaccia tra la scheda dSPACE e la scheda FPGA e tra la scheda analogica e la scheda di conversione Analogico/Digitale (A/D)
7. Scheda di conversione A/D Terasic_THDB_ADA
8. FPGA Altera Cyclone_III Starer Board
9. Motore sincrono a magneti permanenti

Il banco di prova è stato schematizzato in Fig. 2.1. Il software di gestione dell'inverter dSPACE, fornisce i parametri per settare il bus in continua e il controllo del motore. Il bus in continua fornisce la tensione di ingresso all'inverter, il quale alimenta le fasi del motore sincrono a magneti permanenti, le tre tensioni di fase vengono prelevate dall'inverter dSPACE, e iniettate in ingresso alla scheda analogica, la quale in uscita fornisce le due tensioni concatenate scalate (rispetto alle tensioni di fase di ingresso) di un fattore 1:200. Le due tensioni concatenate single ended entrano nella scheda di interaccia, la quale fornisce in uscita le due tensioni concatenate in modo differenziale scalate di un fattore 1:1,5. La scheda di interfaccia inoltre riceve dall'inverter dSPACE il segnale di sincronismo della PWM e lo modifica per renderlo utilizzabile dal software caricato nell'FPGA. Le tensioni concatenate differenziali entrano nella scheda Terasic_THDB_ADA, e vengono convertite in segnali digitali. La scheda FPGA riceve in ingresso, le due concatenate digitalizzate e il segnale di sincronismo PWM opportunamente modificato dalla scheda di interfaccia. Il software caricato nell'FPGA si occupa di ricavare dai segnali di ingresso le corrispondenti fondamentali, ovvero i valori medi dei segnali di ingresso. Le uscite dell'FPGA sono collegate alla scheda Terasic_THDB_ADA che fornirà all'oscilloscopio la conversione analogica dei segnali elaborati dall'FPGA stessa; in modo da ottenere in uscita un segnale proporzionale alla media sul periodo di modulazione PWM della tensione concatenata del motore.

Nei capitoli successivi verranno spiegati in maniera approfondita tutti i componenti hardware che compongono il banco di prova.

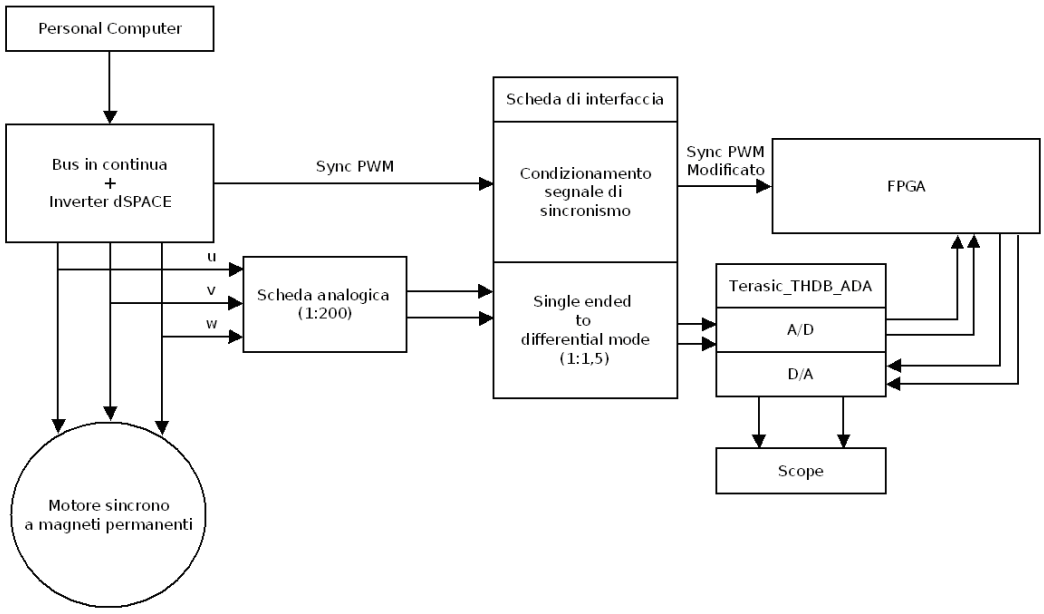

Fig. 2.1: Schema totale hardware utilizzato

2.2 Scheda analogica di acquisizione

Le fasi in ingresso al motore sono state opportunamente condizionate e attenuate utilizzando una scheda analogica. Lo schematico riportato in Fig. 2.2 è stato disegnato utilizzando il kad freeware Kicad, scaricabile dal sito riportato in [4]. Il circuito accetta in ingresso due fasi del motore, e, attraverso una rete di resistenze e un amplificatore operazionale, ottiene in uscita la differenza delle tensioni di fase, ovvero la concatenata, scalata di un fattore 1:200 rispetto ai segnali di ingresso. Nella scheda analogica sono presenti due circuiti speculari, come mostrato in Fig. 2.2; da notare che la presenza del trimmer serve per bilanciare esattamente i due circuiti. Ovvio ricordare che la simmetria tra i due rami del progetto è di fondamentale importanza ai fini di ottenere dei risultati confrontabili, e successivamente utilizzabili per la stima di coppia.

2.3 Scheda di interfaccia

2.3.1 Stadio di trasformazione da Single Ended Mode a Differential Mode

La tensione concatenata single ended scalata di un fattore 1:200 deve essere ora digitalizzata attraverso il convertitore AD della scheda Terasic_THDB_ADA. Tale convertitore accetta segnali in ingresso di tipo differenziale; in particolare su ciascun canale l'AD accetta valori compresi tra -0,5V e +0,5V. Pertanto si è tarato il sistema sul caso peggiore, ovvero immaginando di avere in ingresso una tensione pari a 600Vpp. Quindi, considerando l'attenuazione di 1:200 introdotta dalla scheda analogica; ai fini di interfacciare la tensione concatenata scalata, con il convertitore AD, si deve introdurre uno stadio che attenui di un ulteriore fattore 1:1,5 tale valore di tensione, e che trasformi il segnale da single ended a differenziale. Uno schema a blocchi semplificativo è riportato in Fig. 2.3.

Per permettere al convertitore AD di leggere correttamente le tensioni in ingresso, è stata progettata una scheda di condizionamento delle tensioni concatenate, uscenti dallo stadio analogico di riduzione, il layout della stessa è stato disegnato utilizzando il programma freeware Kicad. La parte della scheda di interfaccia relativa al condizionamento

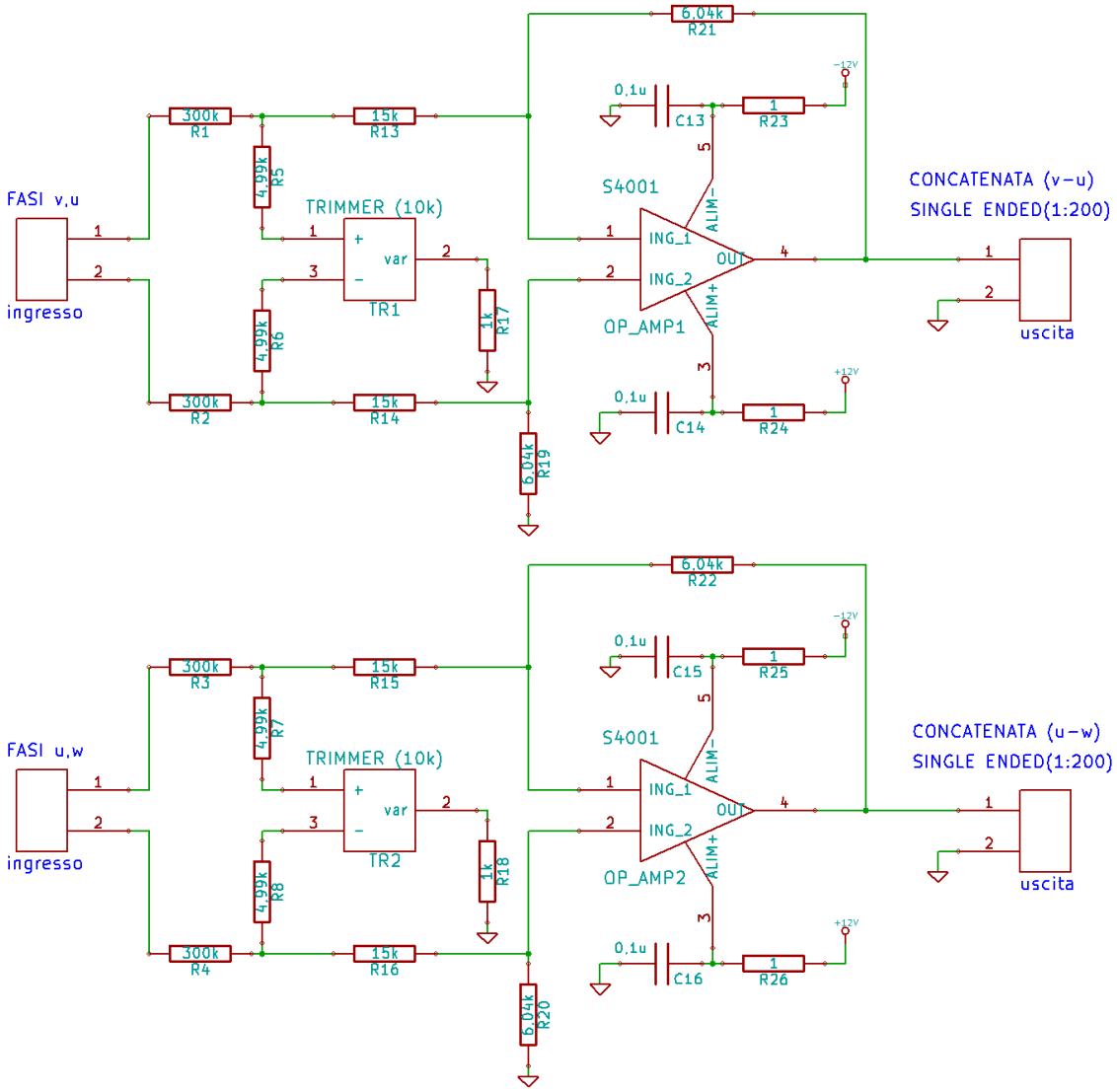

Fig. 2.2: Stadio di attenuazione (scheda analogica)

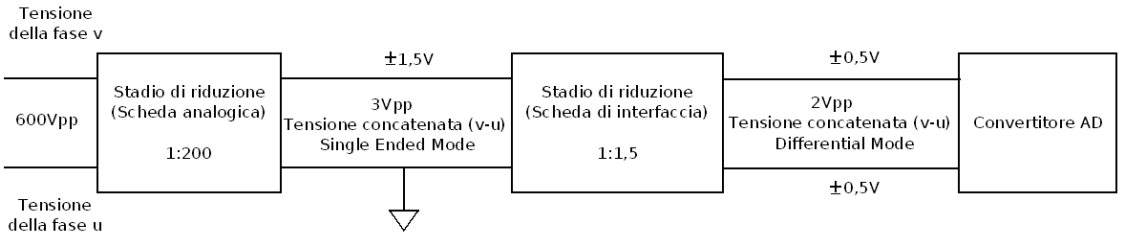

Fig. 2.3: Stadi di attenuazione

delle tensioni concatenate, è costituito da un amplificatore operazionale AD8138 utilizzato per trasformare il segnale da single ended a differenziale, e scalarlo di un fattore 1:1,5 attraverso l'opportuno dimensionamento della rete di resistenze. Lo schematico Kicad è stato riportato in Fig. 2.4.

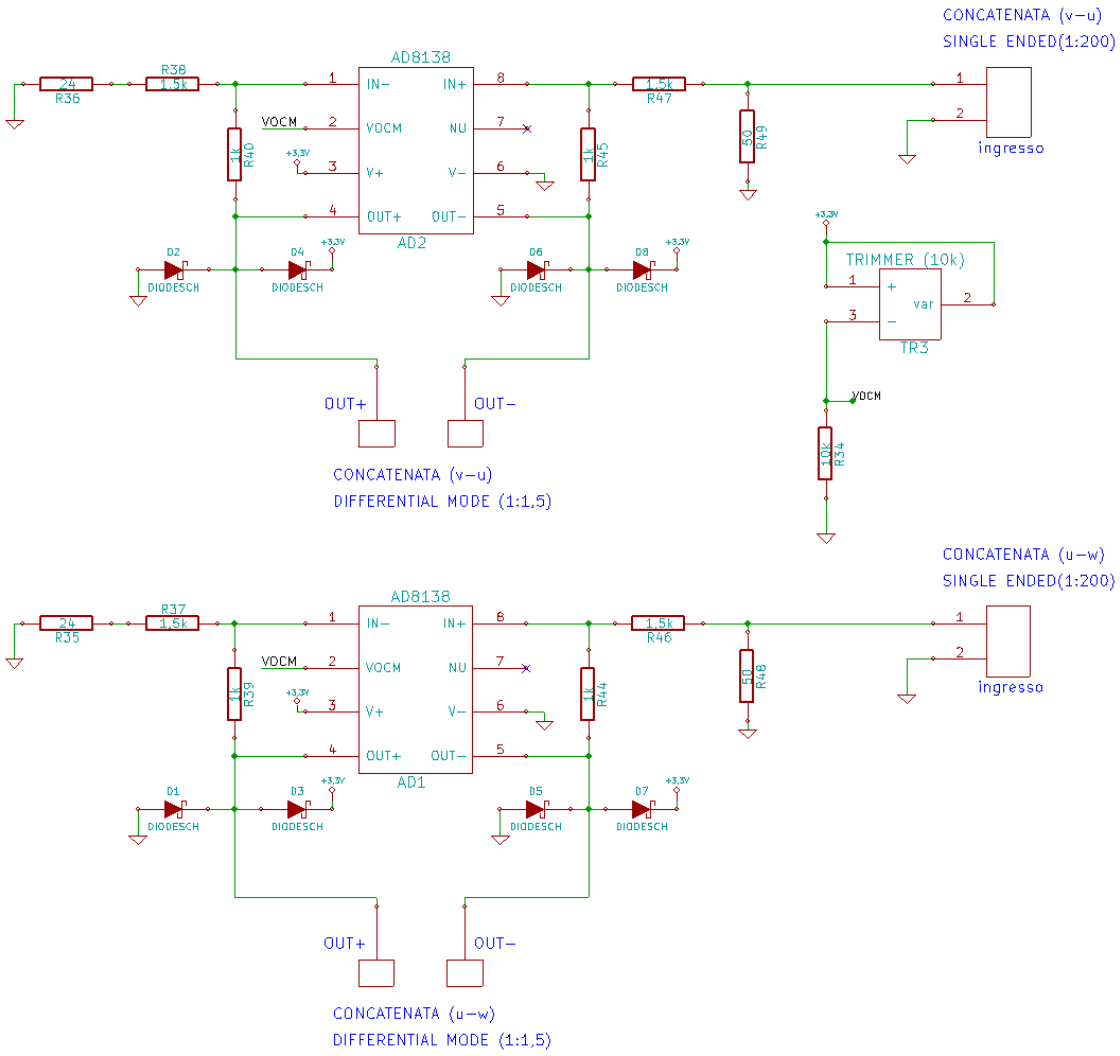

Fig. 2.4: Schematico Kicad: Single Ended to Differential Mode

Il circuito di polarizzazione dell'AD8138 è stato creato seguendo gli esempi relativi alle applicazioni tipiche, che si trovano nel data sheet del componente; per ulteriori approfondimenti si rimanda alla [3]. In Fig. 2.5 viene riportato lo schema a blocchi, estrapolato dal data sheet del componente, e modificato secondo le esigenze di progetto. Da tale schema si possono notare le varie corrispondenze con lo schematico Kicad di Fig. 2.4.

Col fine di ottenere un segnale compatibile con gli ingressi dell'ADC della scheda Terasic_THDB_ADA, il circuito di interfaccia deve fornire un'attenuazione pari a 1:1,5. Il modulo del guadagno dell'amplificatore si ottiene dalla formula 2.1

$$\left|\frac{V_{OUT,dm}}{V_{IN,dm}}\right| = \frac{R_F}{R_G} \tag{2.1}$$

Poiché si vuole ottenere un guadagno pari al reciproco di 1,5; è facile ottenere il dimensionamento delle resistenze R_F e R_G come dalle 2.2

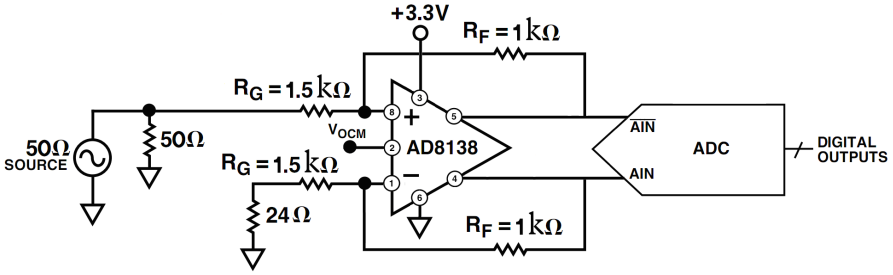

Fig. 2.5: Applicazione tipica AD8138 (Fig. ricavata da [3])

$$R_F = 1k\Omega$$

(2.2)

$$R_G = 1,5k\Omega$$

Si è inoltre dovuto impostare la tensione di modo comune al pin 2 dell'AD8138 (vedi Fig. 2.4 o Fig. 2.5) Tale valore deve corrispondere alla tensione di modo comune del convertitore AD9248 presente nella scheda Terasic_THDB_ADA. La tensione è stata misurata e risulta pari a $1,846V$ (per approfondimenti vedi sottosezione 2.4.2) Per impostare la tensione di modo comune al pin 2; nella scheda progettata in laboratorio (vedi Fig. 2.4), è stato introdotto un partitore resistivo, composto da una resistenza fissa del valore di $10k\Omega$ e un trimmer variabile, tarato e bloccato al valore di $7.876k\Omega$ in maniera da soddisfare la 2.3.

$$V_{OCM} = \frac{R_{FISSA}}{R_{FISSA} + R_{TRIMMER}} V_{cc} = \frac{10k\Omega}{10k\Omega + 7,876k\Omega} 3,3V = 1,846V \tag{2.3}$$

In questo modo il pin 2 dell'AD8138 si porta ad un valore di tensione pari 1,846V, che equivale alla tensione di modo comune del convertitore AD9248.

Col fine di proteggere il convertitore AD9248; nella scheda progettata in laboratorio sono stati introdotti dei diodi Schottky come illustrato in Fig. 2.4. In maniera da limitare i segnali di uscita dall'AD8138 tra il valore di massa e il valore di alimentazione.

Si fa notare infine come la resistenza pari a 24Ω, introdotta in serie al morsetto meno dell'AD8138, serve per ottenere il bilanciamento dei canali di ingresso dell'amplificatore. In particolare essa bilancia il parallelo tra le due resistenze da 50Ω, posto in serie al morsetto positivo dell'AD8138. Il parallelo è composto da una resistenza fisica (introdotta effettivamente nello schematico kicad) e dalla resistenza intrinseca della linea di segnale in ingresso.

2.3.2 Stadio di condizionamento del segnale di sincronismo

La scheda dSPACE fornisce in uscita il segnale di sincronismo della PWM schematizzato in Fig. 2.6. Tale segnale è caratterizzato da un frequenza di 10khz (la frequenza del segnale è settabile a piacere), da un valore logico alto pari a 5V e un valore logico basso pari a 0V.

La scheda FPGA interpreta correttamente segnali in ingresso con valore logico alto compreso tra +1,7V e +4,1V e con valore logico basso compreso tra -05V e +0,7V. E' necessario pertanto condizionare il segnale di sincronismo per renderlo compatibile con l'ingresso dell'FPGA. Oltre a modificare l'ampiezza del segnale allunghiamo anche il tempo

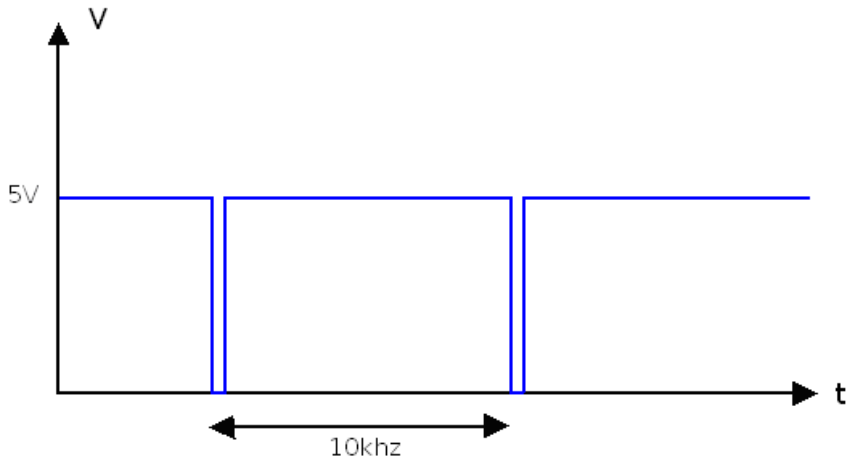

Fig. 2.6: Sincronismo PWM fornito dalla scheda dSPACE
(prima del condizionamento realizzato dalla scheda di interfaccia)

in cui il segnale permane al valore logico basso. In una prima parte del progetto tale modifica era necessaria affinché il software funzionasse correttamente. Ora non lo è più, ma è stata comunque mantenuta in maniera da avere dei fronti di salita e discesa non troppo ravvicinati tra loro, in modo che il software rimanga robusto a fronte di possibili disturbi. Si desidera quindi ottenere un segnale simile a quello riportato in Fig. 2.7. Dove sono stati tratteggiati i limiti di tensione per i valori logici alto (in verde) e basso (in rosso)

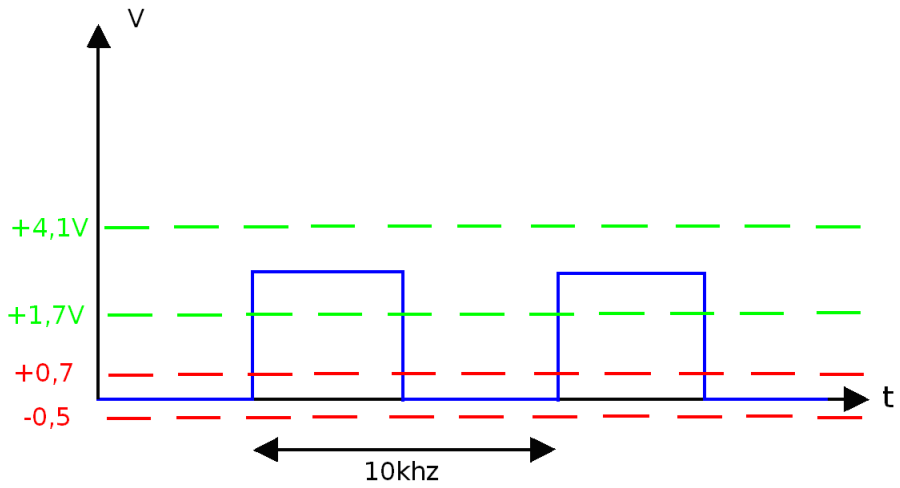

Fig. 2.7: Sincronismo PWM entrante nella scheda FPGA
(dopo il condizionamento realizzato dalla scheda di interfaccia)

Il circuito realizzato ai fini di condizionare il segnale di sincronismo è stato disegnato con il cad freeware Kicad, e riportato in Fig. 2.8.

Il componente monostabile è formato da due parti identiche. Nel progetto è stato utilizzato solamente il lato b del monostabile, i pins del lato a sono stati lasciati flottanti; tranne il pin VSS che è un pin di massa. Per capire il funzionamento di tale circuito riportiamo in Fig. 2.9 la tabella di verità (ricavata dal data sheet del componente) nella quale è riportato il funzionamento logico del dispositivo (per ulteriori approfondimenti si rimanda alla [5])

Si fa notare in particolare il funzionamento indicato nella quarta riga, in quanto è il funzionamento utile al progetto in questione. In tali condizioni si ha l'ingresso A sempre

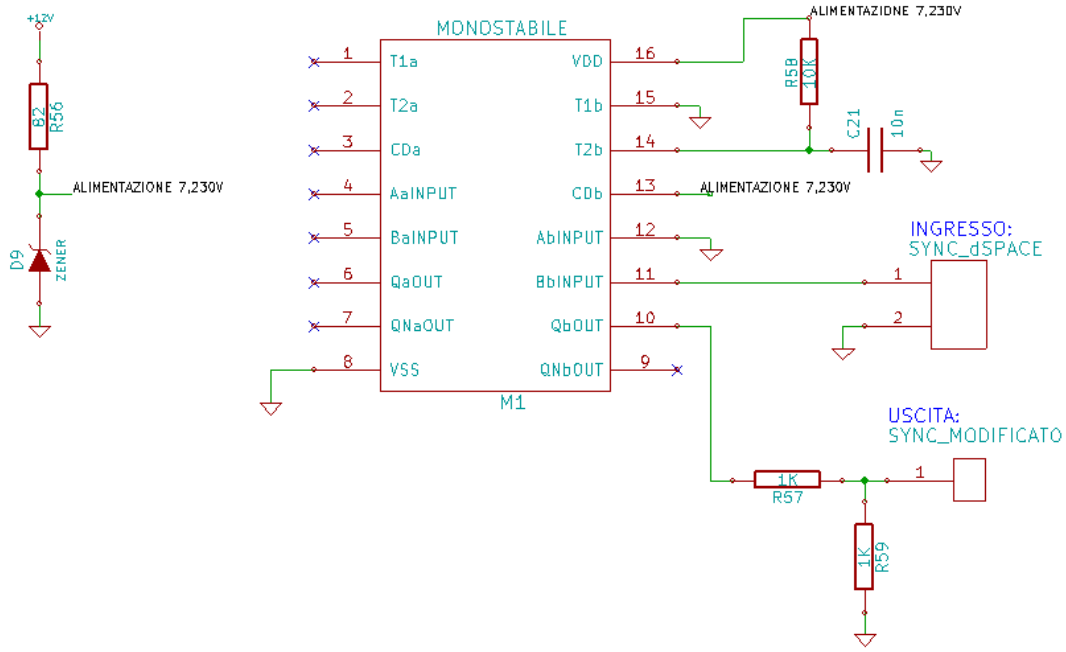

Fig. 2.8: Circuito monostabile (schematico Kicad)

Truth Table

Inputs			Outputs	
Clear	A	B	Q	$\overline{Q}$
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↓	⎍	⎎
H	↑	H	⎍	⎎

H = High Level
L = Low Level
↑ = Transition from Low to High
↓ = Transition from High to Low
⎍ = One High Level Pulse
⎎ = One Low Level Pulse
X = Irrelevant

Fig. 2.9: Funzionamento logico del monostabile

a massa (vedi Fig. 2.8) e l'ingresso B ha un fronte di discesa. In questa condizione, con il segnale di Clear sempre al valore logico alto, si ottiene un impulso positivo dell'uscita Q (e uno negativo nell'uscita Q negata). Nel seguito del capitolo verrà spiegato come sono stati settati la durata e l'ampiezza dell'impulso.

Per comprendere ulteriormente il funzionamento in questione si riporta in Fig. 2.10 l'andamento dei segnali elettrici, ricavato dal data sheet del componente.

Come si può notare dalla Fig. 2.10 il segnale di ingresso A è sempre a massa. In corrispondenza del momento in cui il segnale di ingresso B ha un fronte di discesa, l'uscita Q ha un fronte di salita, e la capacità collegata al pin T2b (vedi Fig. 2.8) comincia a caricarsi a partire da un valore di riferimento interno al monostabile V_{REF1}. Quando il pin T2b raggiunge la tensione di riferimento V_{REF2}, l'uscita Q ha un fronte di discesa. Tale andamento si ripete nel tempo. Di seguito si spiegheranno in maniera più approfondita le scelte adottate sul circuito in esame.

Si è ricavato dal data sheet come la tensione di alimentazione del monostabile (ossia la tensione fornita al pin VDD, vedi Fig. 2.8) influenzi la sensibilità del componente ai valori logici in ingresso, l'ampiezza e la durata dell'impulso positivo d'uscita. In ingresso al monostabile il segnale di sincronismo ha valore alto pari a 5V e valore basso pari

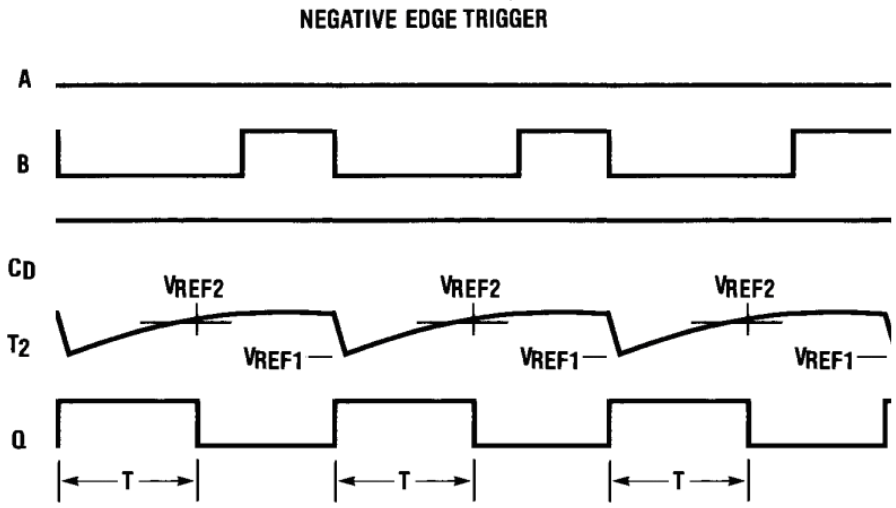

Fig. 2.10: Andamento dei segnali all'interno del dispositivo

a 0V. Pertanto, per essere sensibili a tali valori di ingresso, si è inserito nel circuito un diodo zener polarizzato inversamente (vedi Fig. 2.8), in grado di stabilizzare la tensione di alimentazione al monostabile al valore di circa 7,23V. Con tale valore di alimentazione il componente interpreta correttamente segnali di ingresso (considerando il valore tipico a 25°C) con valore logico basso fino a circa 3.2V e valore logico alto a partire da circa 4V Quindi il segnale proveniente dalla dSPACE è perfettamente dentro al range degli ingressi accettati dal monostabile. Per ottenere l'alimentazione desiderata, è stata dimensionata la resistenza di polarizzazione dello zener in modo da ottenere una tensione sufficientemente stabile. Per far questo è stato scelto un punto di lavoro relativamente lontano dal gomito del breakdown. E' stato ottenuto un punto di lavoro con le seguenti caratteristiche:

Tensione di alimentazione (proveniente dall'FPGA) pari a:

$$V_{DD} = 11,396V \tag{2.4}$$

Resistenza:

$$R_{56} = 82\Omega \tag{2.5}$$

Tensione di breakdown:

$$V_{bd} = 7,23V \tag{2.6}$$

Corrente di break down:

$$Ibd = \frac{11,396V - 7,230V}{82\Omega} = 50,8mA \tag{2.7}$$

Potenza dissipata sullo zener (il diodo sopporta fino a 0,5W):

$$7,23V \cdot 50,8mA = 0,376W \tag{2.8}$$

Si è precedentemente detto in questa sottosezione che l'alimentazione del monostabile influenza l'ampiezza dell'impulso di uscita, infatti, alimentando il componente a circa 7,230V il valore alto dell'uscita è pari a circa 6,4V. Quindi, ai fini di rimanere dentro al range di tensioni di ingresso accettate dall'FPGA, (valore logico alto compreso tra 1,7V e 4,1V; valore logico basso compreso tra -0,5V e 0,7V) è stato introdotto nella scheda di

interfaccia (vedi Fig. 2.8) un partitore resistivo collegato all'uscita Qb (PIN 10).

Il partitore è formato dalle resistenze:

$$R_{57} = 1k\Omega \qquad R_{59} = 1k\Omega \tag{2.9}$$

in modo da ottenere un'impulso d'uscita con valore logico alto pari a:

$$6,4V \cdot \frac{R_{59}}{R_{57} + R_{59}} = 3,2V \tag{2.10}$$

Quando il pin d'uscita del monostabile viene collegato in ingresso all'FPGA il valore logico alto viene bloccato a 3V mentre il valore logico basso rimane a 0V.

Infine si illustra la procedura per ottenere la larghezza dell'impulso d'uscita desiderata. La dSPACE fornisce un segnale di sincronismo con frequenza pari a 10khz e forma d'onda riportata in Fig. 2.11. Si desidera ottenere un segnale con duty cycle nell'intorno del 50%. Pertanto la larghezza dell'impulso d'uscita desiderata è nell'intorno dei $50\mu s$. Dal data sheet del componente vediamo che con una capacita pari a 10nF (collegata tra i pin T2b e T1b di Fig. 2.8) e una resistenza pari a $10k\Omega$ (collegata tra i pin VDD e T2b di Fig. 2.8), si ottiene un valore tipico di ampiezza dell'impulso pari a $30\mu s$ se VDD = 5V; a $50\mu s$ se VDD = 10V. Pertanto, utilizzando tali valori di capacità e resistenza e alimentando nell'intorno dei 7V, ci si trova in una situazione intermedia, ottima ai fini del progetto. Per una stima iniziale della larghezza dell'impulso si è utilizzata anche la formula teorica, ricavata dal data sheet del componente:

$$t_W = 0,2R_X C_X \cdot Ln\left[V_{DD} - V_{SS}\right] \tag{2.11}$$

Considerando R_X in Ohm, C_X in Farad, V_{DD} e V_{SS} in Volts, si ottiene un impulso di durata pari a $39,6\mu s$. Il valore reale dell'impulso, misurato in laboratorio, è pari a $43\mu s$, sufficientemente vicino a quello previsto dall'analisi teorica, e perfettamente aderente alle esigenze di progetto. Si riporta in Fig. 2.12 il risultato, ottenuto con Matlab, del processamento di un milione di campioni, estratti dalle misurazioni effettuate con l'oscilloscopio.

2.3.3 Progettazione della scheda di interfaccia

Nelle sottosezioni 2.3.1 e 2.3.2 sono stati spiegati in maniera approfondita i due circuiti che compongono la scheda di interfaccia, in particolare il circuito di trasformazione del segnale di ingresso da single ended a differenziale, e il circuito di condizionamento del segnale di sincronismo. Di seguito si approfondiranno le regole generali adottate ai fini della realizzazione della scheda. Lo schema elettrico totale della scheda, realizzato tramite il software freeware Kicad, è riportato in Fig. 2.13.

Dallo schema circuitale si ottiene il disegno della basetta come riportato nelle Fig. 2.14, Fig. 2.15 e Fig. 2.16. Tali figure sono ottenute utilizzando il software freeware Wings 3D, abbinato al Kicad.

La basetta creata in laboratorio è dotata di due strati di rame, separati da uno strato isolante in vetronite. In Fig. 2.14 si possono notare le piste di rame, disegnate in rosso,

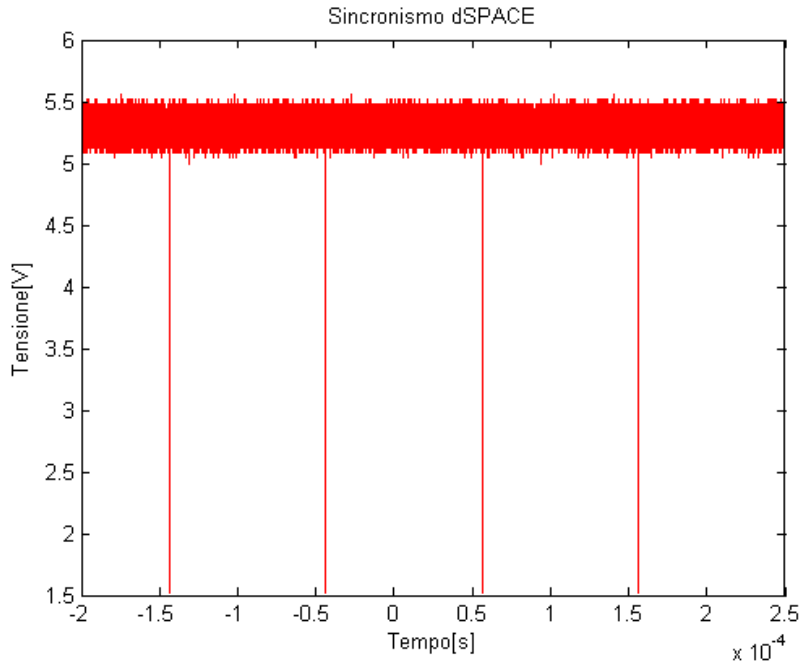

Fig. 2.11: Sincronismo dSPACE

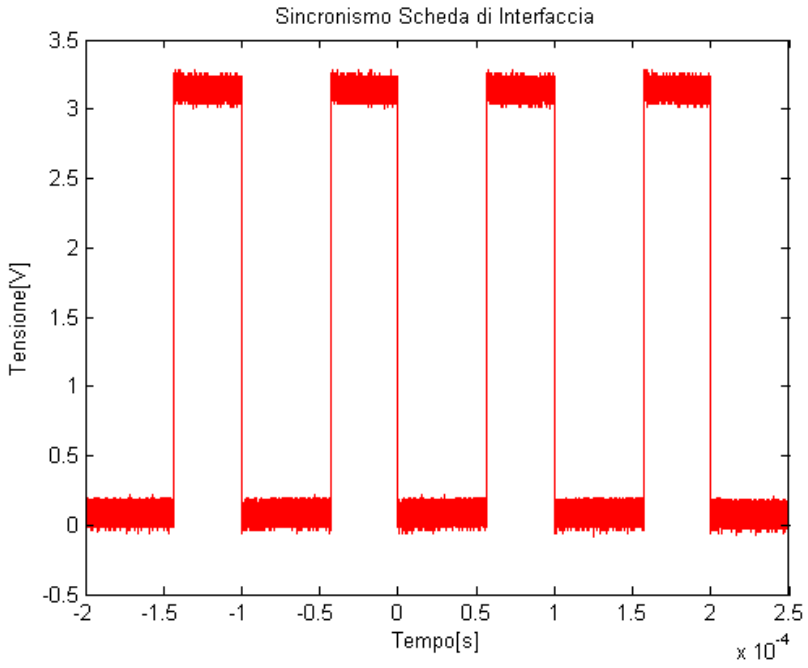

Fig. 2.12: Sincronismo scheda di interfaccia

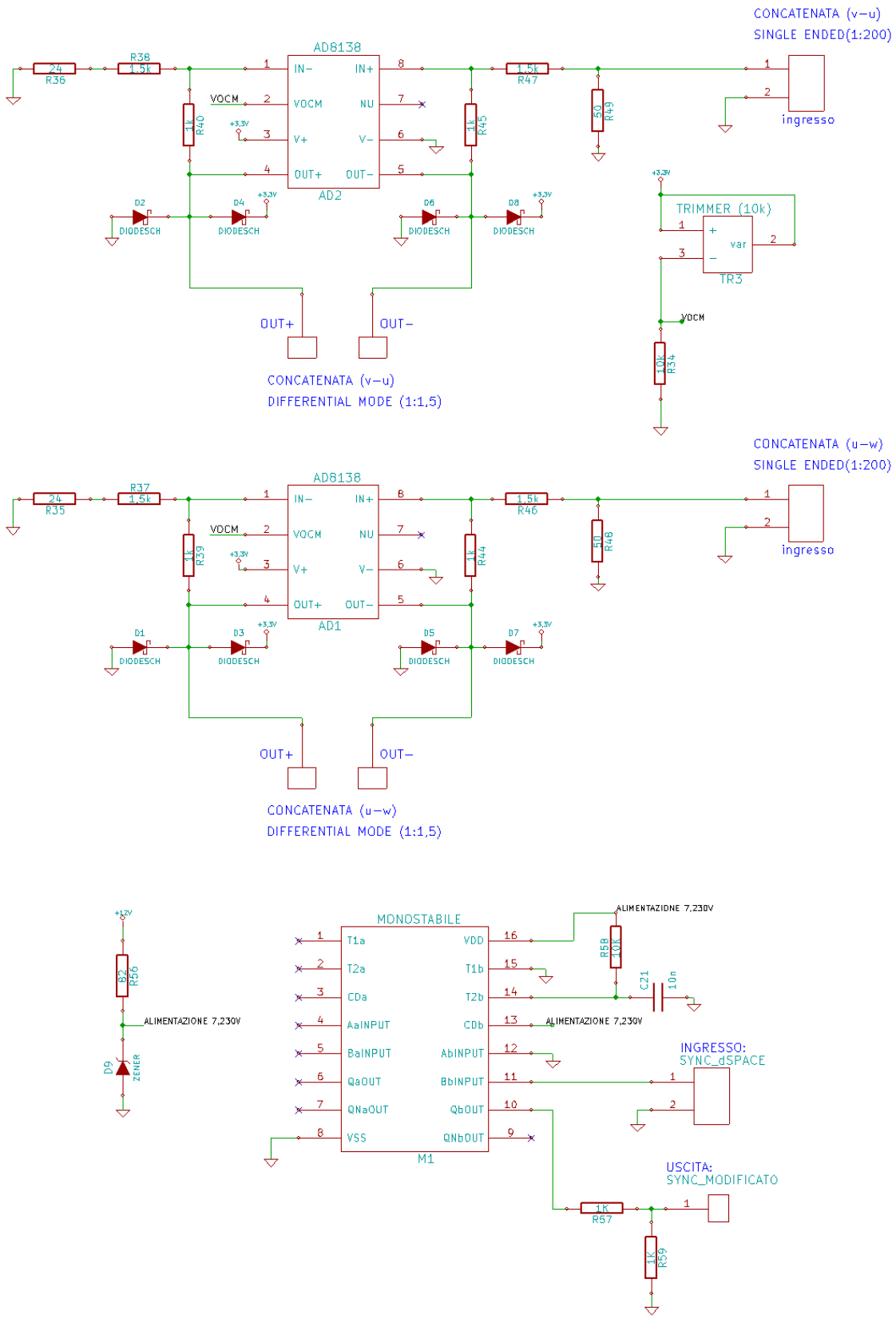

Fig. 2.13: Kicad: Schema circuitale scheda di interfaccia

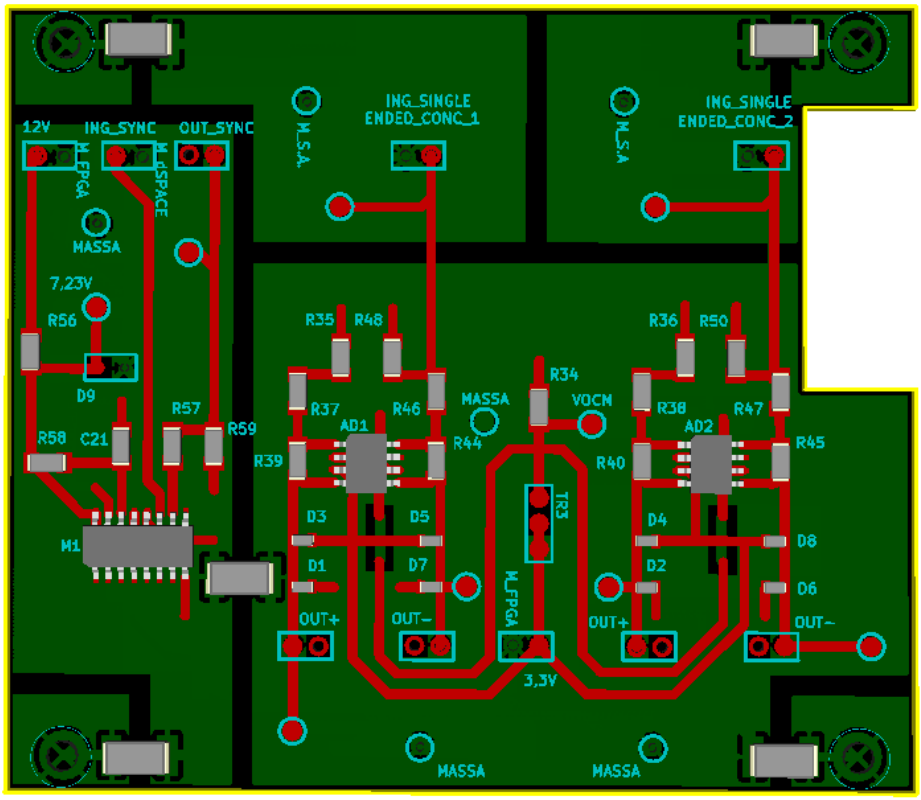

Fig. 2.14: Wings 3D: Basetta vista faccia superiore

poste nella faccia superiore della scheda. Si è scelto di far passare nello strato superiore tutte le piste di segnale e di alimentazione. La parte della scheda dedicata alla trasformazione del segnale in ingresso da single-ended a differenziale è composta da due circuiti idealmente uguali. Per ottenere nella pratica questo risultato si è cercato di realizzare un layout il più simmetrico possibile, in modo che segnali corrispondenti nei due circuiti compiano percorsi duali. Inoltre si è cercato di utilizzare, resistenze con valori il più uguali possibili, sempre per ottenere una notevole dualità tra i due circuiti. Le piste dei due circuiti sono affiancate e hanno una notevole simmetria per ottenere una maggiore robustezza a fronte di possibili disturbi irradiati. In modo che un'ipotetico disturbo colpisca nello stesso modo piste corrispondenti nei due circuiti simmetrici. Con questa accortezza lo stesso disturbo si accoppierà nello stesso modo in entrambi i circuiti, quindi il suo effetto sarà visibile allo stesso modo in entrambi i canali. Così facendo si evita di ottenere due risultati diversi in uscita a causa dell'accoppiamento di un disturbo solo su uno dei due canali della scheda. I segnali di ingresso e uscita sono onde quadre a 10khz, essendo segnali con fronti ripidi, sono formati da armoniche ad alta frequenza, che potrebbero causare delle emissioni irradiate. Quindi si è cercato, per quanto possibile, di tenere distanziati i segnali di ingresso da quelli di uscita, per evitare che gli uni si accoppiassero agli altri e viceversa.

In Fig. 2.15 si può notare la faccia inferiore della scheda di interfaccia, dove si osserva una presenza abbondante di rame. Infatti la faccia inferiore della scheda è stata scelta come piano di massa. Si vedono bene anche i fossati, di colore nero, per la divisione dei piani di massa. La scelta di dedicare un piano intero ai percorsi di massa, anziché utilizzare delle piste di rame, è stata adottata col fine di ottenere una maggiore immunità ai disturbi. Infatti, in questo modo, le correnti di ritorno dei vari segnali scelgono il percorso più favorevole, non andando a disturbare altri segnali. In partiolare il percorso

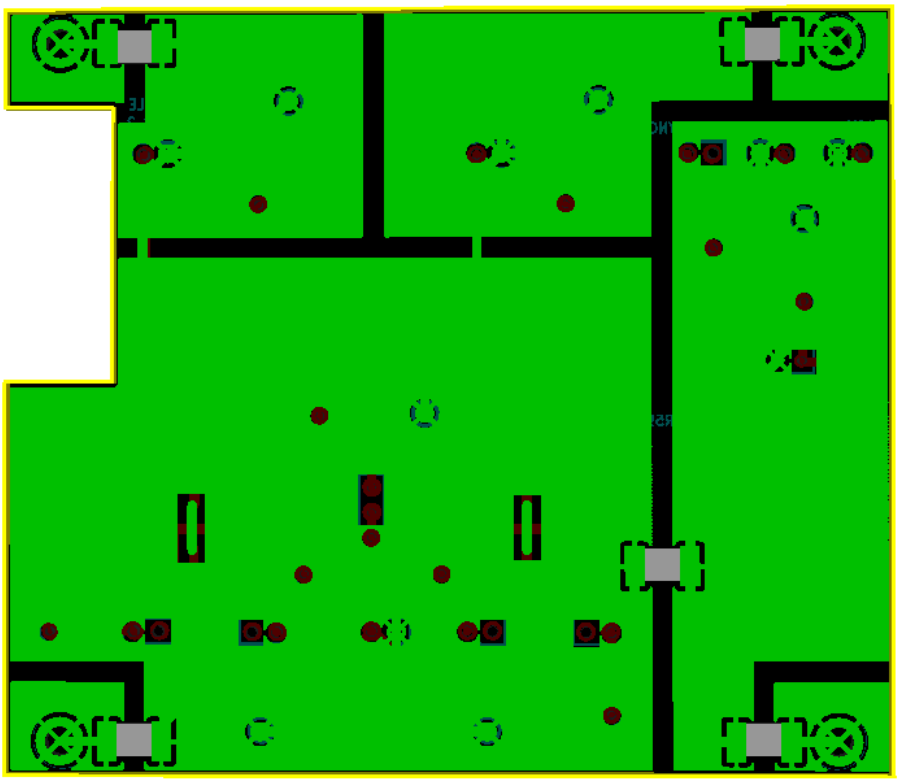

Fig. 2.15: Wings 3D: Basetta vista faccia inferiore

più favorevole per una corrente di ritorno risulta essere quello sotto alla pista di segnale a cui corrisponde tale corrente. In questo modo le correnti percorrono meno spazio possibile, non andando a creare degli ampi loop che potrebbero essere causa di emissioni elettromagnetiche, che potrebbero disturbare l'intero circuito. Col fine di non incrociare percorsi di massa, provenienti da masse differenti, sono stati introdotti, come già citato sopra dei fossati, ovvero degli spazi sul rame. In particolare il piano di massa del circuito monostabile e il piano di massa del circuito per la trasformazione del segnale di ingresso da single-ended a differenziale non hanno un collegamento fisico sul piano di massa della scheda. Mentre, i piani di massa relativi all'ingresso delle due concatenate sono uniti al piano di massa del circuito per la trasformazione del segnale di ingresso da single-ended a differenziale, da una pista di rame, di piccola sezione, posta esattamente sotto la pista di segnale. In questo modo tra i tre piani di massa scorreranno soltanto le correnti di ritorno delle concatenate in ingresso alla scheda, e non altri segnali che potrebbero introdurre dei disturbi sui segnali di ingresso.

Dalla Fig. 2.16 si può avere una visione in tre dimensioni della scheda, grazie alla quale si possono maggiormente distinguere, lo strato superiore delle piste di segnale e il piano di massa inferiore, inoltre sono ben evidenti tutti i fori che servono per collegare il piano di massa posto sullo strato inferiore della scheda con le piste di massa che si trovano sulla parte superiore della scheda. Si riporta in Fig. 2.17 una foto della scheda di interfaccia realizzata in laboratorio, posizionata sopra la scheda di conversione analogica/digitale, a sua volta collegata alla scheda FPGA.

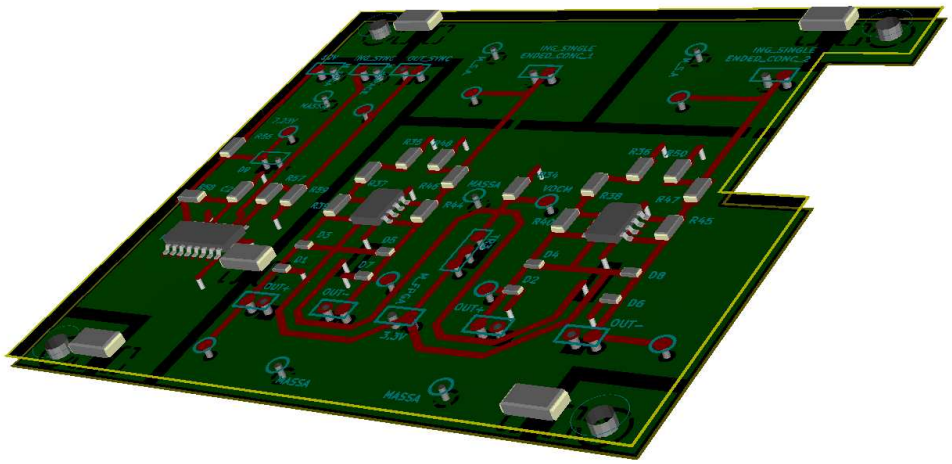

Fig. 2.16: Wings 3D: Basetta 3D

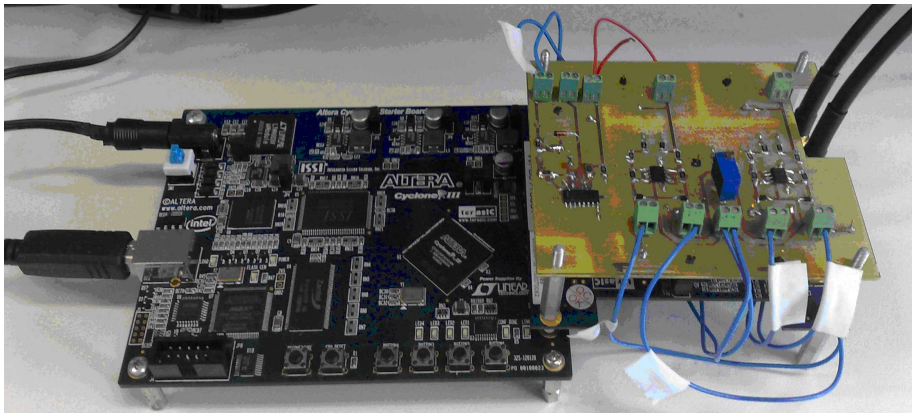

Fig. 2.17: Scheda di interfaccia (collegata al sistema FPGA + ADA Converter)

2.4 Scheda di acquisizione analogica digitale

2.4.1 Introduzione

La scheda di acquisizione analogica digitale Terasic_THDB_ADA, viene utilizzata nel progetto, a monte dell'FPGA per trasformare in digitale i segnali analogici provenienti dalla scheda di interfaccia, e a valle dell'FPGA per trasformare il risultato dell'elaborazione dell'FPGA da digitale ad analogico, col fine di poterlo visualizzare correttamente sull'oscilloscopio. Per poter realizzare quanto sopra descritto la scheda Terasic_THDB_ADA è dotata di due convertitori AD e due convertitori DA, le cui caratteristiche saranno approfondite nelle sottosezioni 2.4.2 e 2.4.3. In Fig. 2.18 si riporta una foto della scheda di aquisizione analogica digitale. La scheda Terasic_THDB_ADA si collega alla scheda FPGA_Cyclone_III_Starter_Board tramite il connettore HSMC apposito. Si riporta in Fig. 2.19 una foto delle due schede collegate tra loro.

2.4.2 Il convertitore AD

Il segnale opportunamente scalato e reso di tipo differenziale, può essere correttamente letto dal convertitore AD della scheda Terasic_THDB_ADA. Il convertitore AD trasforma la tensione in ingresso in un segnale digitale, che viene poi letto dal programma precaricato nella scheda FPGA.

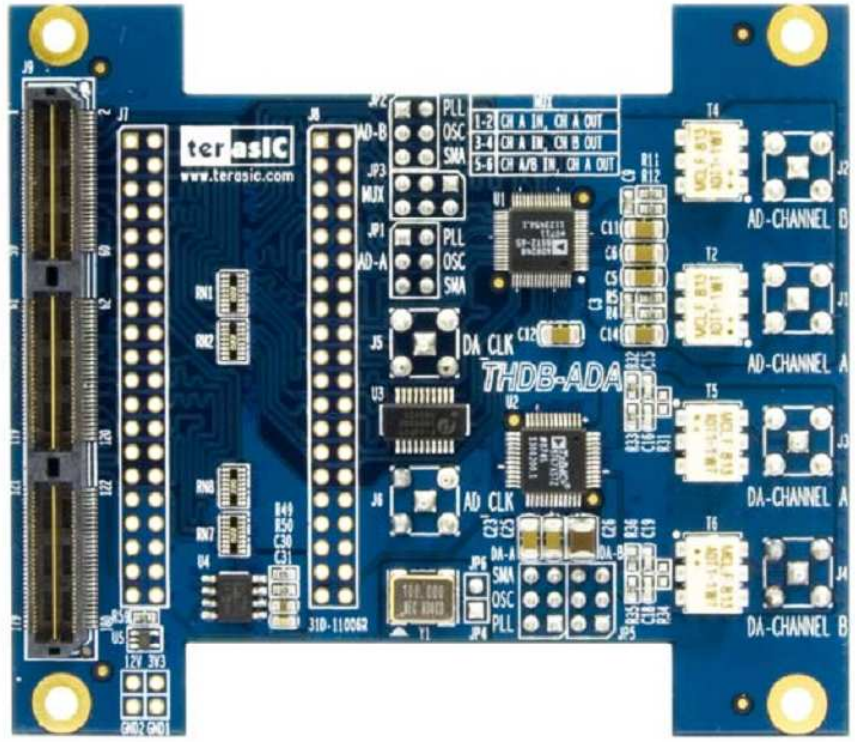

Fig. 2.18: Terasic_THDB_ADA

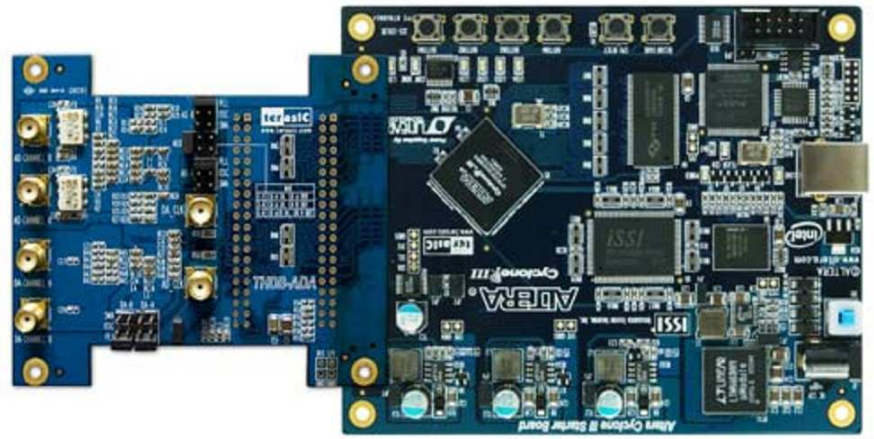

Fig. 2.19: Collegamento tra Terasic_THDB_ADA e FPGA_Cyclone_III_Starter_Board

Il convertitore AD9248 è dotato delle seguenti caratteristiche principali:

1. due AD con profondità di 14bits
2. due ingressi analogici, con span di ingresso compreso tra 1Vpp e 2Vpp
3. due uscite digitali offset binary (o complemento a due)
4. velocità limite di campionamento pari a 65MSPS (Mega Sample Per Second)

In Fig. 2.20 si riporta il diagramma funzionale del convertitore AD9248 compreso di pin-out.

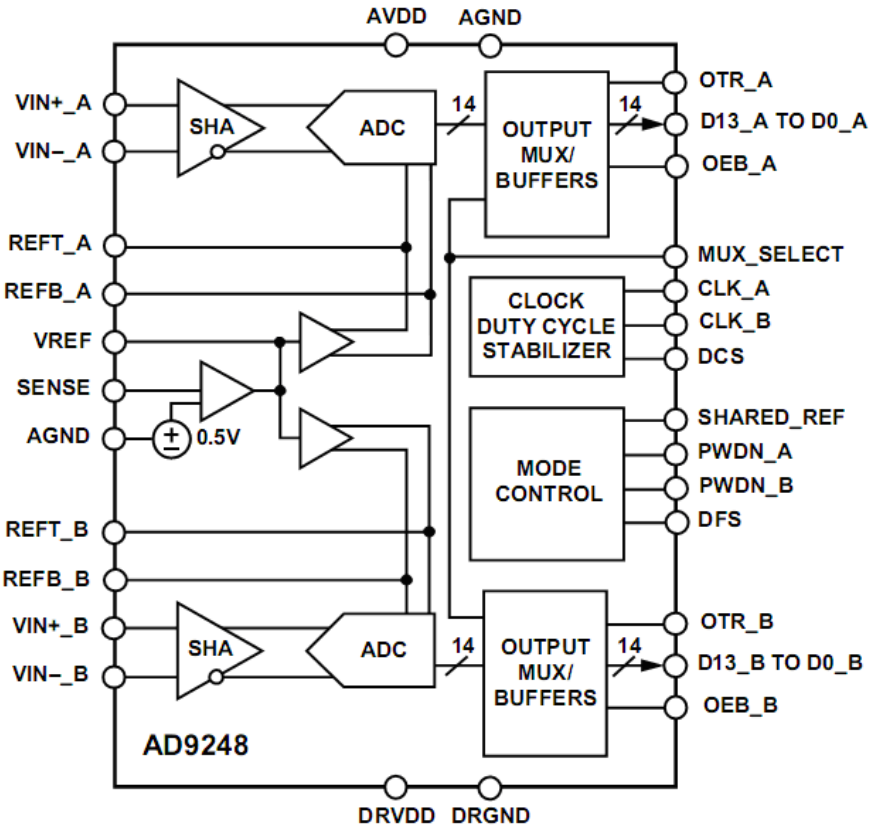

Fig. 2.20: AD9248, diagramma funzionale

Nel seguito del capitolo si analizzerà il funzionamento del convertitore AD9248, per ulteriori dettagli si rimanda alla [6]. L'AD9248 consiste in due ADCs che sono basati sul convertitore AD9235. Gli ingressi analogici dell'AD9248 sono di tipo differenziale; ciascun canale del segnale differenziale, entra nella capacità di switching del sample and hold (vedi Fig. 2.21) creata per offrire le massime performance con segnali differenziali.

Il segnale di clock alternativamente comanda il SAH tra il sample mode e l'hold mode. Quando il SAH è in sample mode, il segnale sorgente carica le capacità di sample; (5pF) la tensione ai suoi capi si assesta entro mezzo ciclo di clock. Un buffer di riferimento interno differenziale crea due riferimenti di tensione, uno positivo REFT e uno negativo REFB, che definiscono lo SPAN interno dell'ADC L'uscita di modo comune del buffer di riferimento è settata a metà dell'alimentazione. Cioè dovrebbe essere settata a 3,3V/2=1.65V Le misurazioni in laboratorio si discostano dall'idealità teorica, come verrà evidenziato

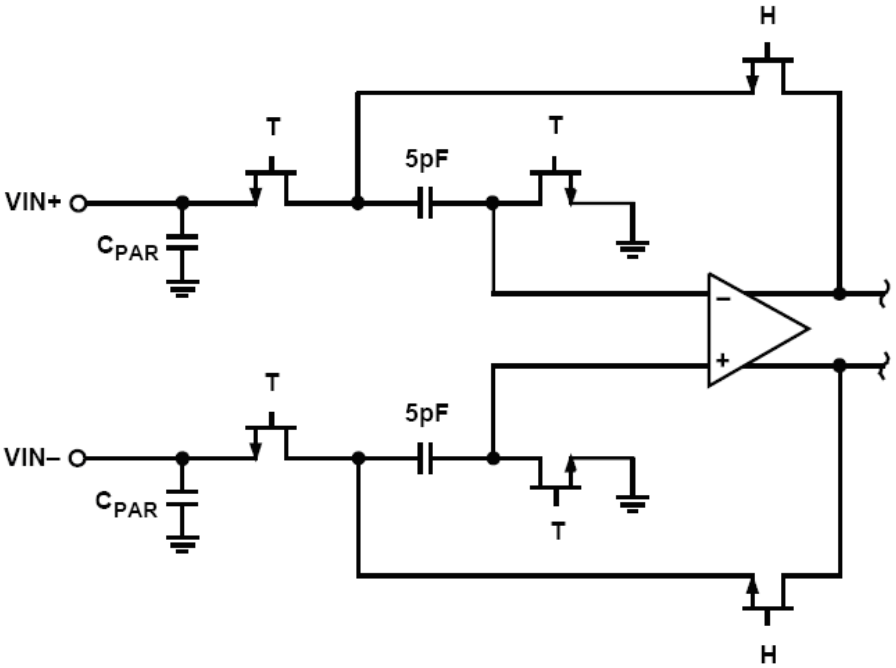

Fig. 2.21: AD 9248, Blocco Sample And Hold

successivamente in questo capitolo. I valori di $REFT$, $REFB$, $SPAN$ e V_{OCM} ideali sono dati rispettivamente dalle 2.14, 2.15, 2.16 e 2.17.

$$AVDD = 3,3V \tag{2.12}$$

$$V_{REF} = 1V \tag{2.13}$$

$$REFT = \frac{1}{2}(AVDD + V_{REF}) = 2,15V \tag{2.14}$$

$$REFB = \frac{1}{2}(AVDD - V_{REF}) = 1,15V \tag{2.15}$$

$$SPAN = 2(REFT - REFB) = 2V_{REF} = 2V \tag{2.16}$$

$$V_{OCM} = V_{MED} = \frac{1}{2}(REFT + REFB) = 1,65V(= \frac{AVDD}{2}) \tag{2.17}$$

Le equazioni mostrano che i valori di tensione $REFT$ e $REFB$ sono simmetrici rispetto a metà dell'alimentazione e lo $SPAN$ è pari a due volte il valore di tensione V_{REF}. Queste formule danno l'idea del funzionamento dell'ADC ma, misurando con l'oscilloscopio i valore reali, si è notato che essi si discostano da quelli ideali. Si riportano nelle 2.18, 2.19, 2.20, 2.21, 2.22 le misurazioni ottenute.

$$V_{REF} = 1,205V \tag{2.18}$$

$$REFT\_A = 2,358V \tag{2.19}$$

$$REFB\_A = 1,358V \tag{2.20}$$

$$REFT\_B = 2,334V \tag{2.21}$$

$$REFB\_B = 1,334V \tag{2.22}$$

Nelle suddette equazioni sono stati riportati i valori di REFT e $REFB$ a seconda del canale di riferimento A e B. Come precedentemente spiegato nela sottosezione 2.3.1; ai fini di collegare in maniera corretta la scheda di interfaccia (creata in laboratorio) con la scheda di acquisizione analogica/digitale si ha bisogno di un riferimento V_{OCM} unico. Quindi il valore di tensione V_{OCM}, dato dalla 2.25, è ottenuto dalla media tra i due valori di modo comune dei due canali, risultanti dalle 2.23 e 2.24.

$$V_{OCM\_A} = \frac{1}{2}(REFT\_A + REFB\_A) = 1,858V \tag{2.23}$$

$$V_{OCM\_B} = \frac{1}{2}(REFT\_B + REFB\_B) = 1,834V \tag{2.24}$$

$$V_{OCM} = \frac{1}{2}(V_{OCM\_A} + V_{OCM\_B}) = 1,846V \tag{2.25}$$

Da notare come il valore ottenuto dalla media tra i due canali (1,846V) si discosti dal valore ideale (1,65V).

Per proseguire nella trattazione è utile riportare lo schema circuitale dell'AD9248 e dell'interfacciamento con la scheda Terasic_THDB_ADA. Lo schematico del convertitore AD9248 insieme al circuito di comando è stato estrapolato da [7], ed è stato ridisegnato e modificato, a seconda delle esigenze di progetto, con il Kicad e riportato in Fig. 2.22.

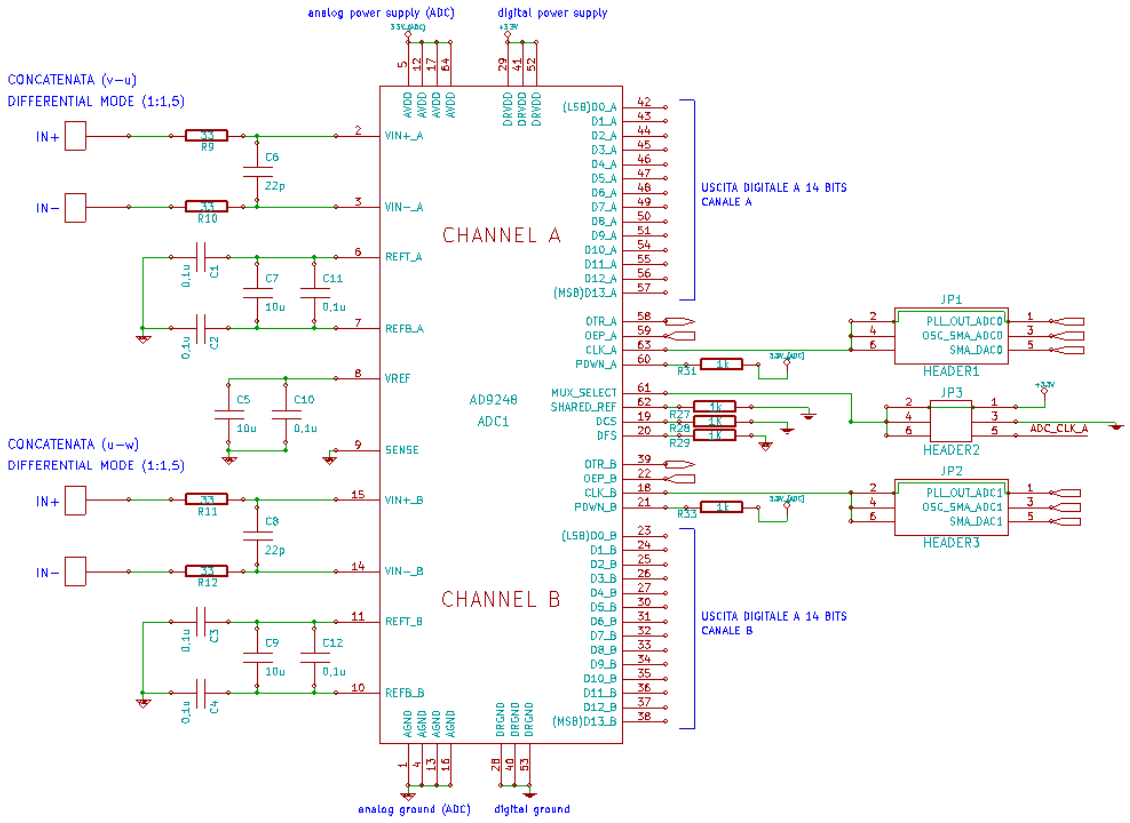

Fig. 2.22: AD 9248, schema circuitale

La parte analogica e quella digitale hanno due alimentazioni separate che, nella scheda Terasic_THDB_ADA, lavorano entrambe a 3,3V. L'AD9248 ha al suo interno due stabilizzatori di duty-cycle, uno per ciascuno dei due ADCs. Se il pin DCS viene forzato al valore logico alto, vengono abilitati gli stabilizzatori interni a fornire alla circuiteria dell'AD9248 un duty cycle stabile al 50%. Questa caratteristica non viene sfruttata nella Terasic_THDB_ADA perché il pin DCS è collegato a massa attraverso una resistenza da $1k\Omega$ (vedi ingrandimento di Fig. 2.23)

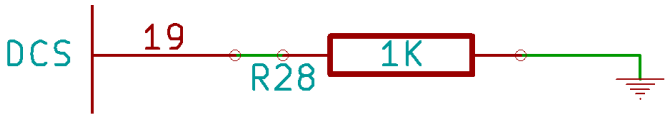

Fig. 2.23: pin DCS

Le alimentazioni per i drivers dei clock degli ADCs sono separate dai drivers delle uscite degli ADCs, per evitare di modulare i segnali di clock con rumore digitale. I clock ai due ADCs nel progetto vengono dati via software; attraverso un PLL (vedi sottosezioni 3.5.1 e 3.5.2) che, partendo dal contatore interno a 50MHz della FPGA, fornisce il clock desiderato. Dal punto di vista circuitale, per forzare i clock dei due ADCs al valore creato via software, si deve applicare un jumper che connetta i terminali 1 e 2 dei connettori JP1 e JP2, come illustrato in Fig. 2.24.

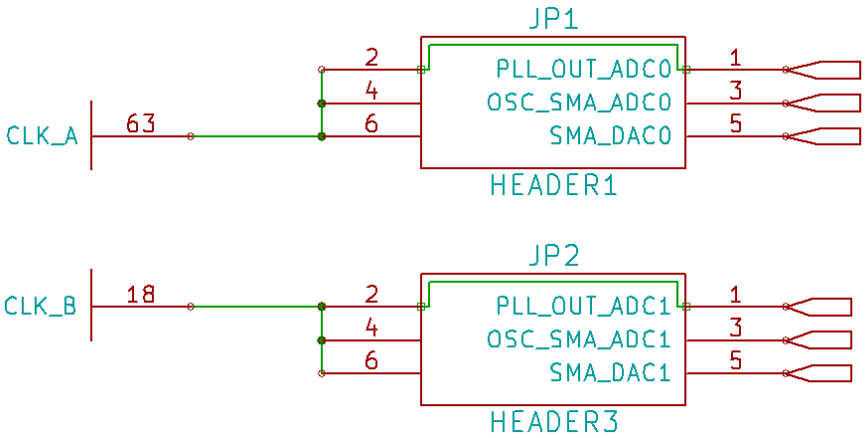

Fig. 2.24: pin CLK_A e pin CLK_B

I terminali collegati ai pin OSC_SMA_ADC0/1 e SMA_DAC0/1 permetterebbero di utilizzare un clock esterno alla scheda, opportunamente collegato attraverso i jack sma (SubMiniature version A) appositi. Nel progetto tale possibilità non è stata utilizzata. Collegando il pin DFS a massa tramite un resistenza da $1k\Omega$, come riportato in Fig. 2.25 si sceglie di utilizzare, per i dati di uscita dall'ADC, il formato Offset Binary (vedi sottosezione 3.4.3)

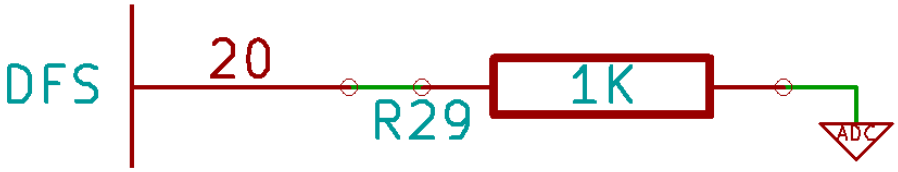

Fig. 2.25: pin DFS

I pin OEP_A e OEP_B (Output Enable Pin) servono per abilitare i canali di uscita dei due ADCs. Per far questo vengono settati al valore logico basso via software (vedi sottosezione 3.5.3). Il valore di tensione basso viene riportato ai due pin dall'FPGA attraverso il connettore HSMC (vedi sezione 2.5). Il pin OTR_A (Out Of Range Channel A) e il pin OTR_B (Out Of Range Channel B) sono pin di uscita degli ADCs, che servono ad indicare se in ingresso ai due ADCs si ha un valore analogico che supera i limiti di tensione accettati dal convertitore AD. I pin PDWN_A e PDWN_B permettono di porre i canali di uscita dei due ADCs in Standby. In maniera da consumare poca potenza, quando non è richiesta alcuna elaborazione da parte dell'ADC. Nella Terasic_THDB_ADA i pin sono collegati al valore logico alto abilitando i canali di uscita dei due ADCs, come riportato in Fig. 2.26.

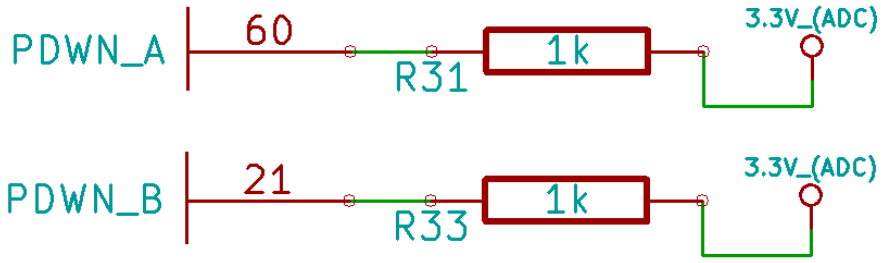

Fig. 2.26: pin PDWN_A, pin PDWN_B

Una volta convertiti in digitale, i dati di uscita escono nel modo seguente. Se il pin MUX_SELECT è settato al valore logico alto, i dati del canale A escono nel bus di uscita a 14bits del canale A; i dati del canale B escono nel bus di uscita a 14bits del canale B. Se il pin MUX_SELECT è settato al valore logico basso, viceversa. Se il pin MUX_SELECT viene comandato in ingresso da un certo clock, i dati rimbalzano tra un canale e l'altro. Nel progetto è stata scelta la prima configurazione; per far questo si imposta un jumper tra i terminali 1 e 2 del connettore JP3 nella scheda Terasic_THDB_ADA (come riportato in figurename 2.27) in modo da collegare il pin MUX_SELECT al valore dell'alimentazione 3,3V.

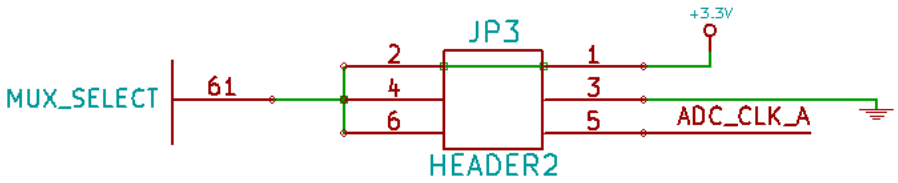

Fig. 2.27: pin MUX_SELECT

Lo shared reference mode permette all'utilizzatore di connettere i riferimenti dei due ADCs entrambi all'esterno, per maggiori performance. Se si intende far funzionare gli ADCs in maniera indipendente, il disaccoppiamento dei riferimenti interni di tensione (dei due ADCs) può essere trattato indipendentemente, e può fornire un maggior isolamento tra i due canali. Per abilitare lo shared reference mode, il pin SHARED_REF deve essere collegato al valore logico alto, e i riferimenti esterni devono essere esternamente cortocircuitati. REFT_A deve essere esternamente cortocircuitato a REFT_B, e REFB_A deve essere cortocircuitato a REFB_B. Nel progetto non vengono utilizzati riferimenti esterni (viene bensì utilizzato il riferimenti interno pari a circa 1V) pertanto non si fa uso dello shared reference mode, e quindi il pin SHARED_REF è collegato a massa attraverso una resistenza da $1k\Omega$, come in Fig. 2.28.

Fig. 2.28: pin SHARED_REF

Un comparatore interno all'AD9248 rileva il potenziale al pin SENSE, e configura il riferimento (V_{REF}) in quattro possibili stati, che sono sintetizzati nella tabella di Fig. 2.29:

Selected Mode	SENSE Voltage	Resulting V_{REF} (V)	Resulting Differential Span (V p-p)
External Reference	AVDD	N/A	2 × External Reference
Internal Fixed Reference	V_{REF}	0.5	1.0
Programmable Reference	0.2 V to V_{REF}	0.5 × (1 + R2/R1)	2 × V_{REF}
Internal Fixed Reference	AGND to 0.2 V	1.0	2.0

Fig. 2.29: Riferimento di tensione

Nella scheda Terasic_THDB_ADA, il pin SENSE è collegato a massa e il pin VREF è collegato a due capacità in parallelo, come in Fig. 2.30. Quindi ci si trova nella situazione schematizzata nella quarta riga di Fig. 2.29.

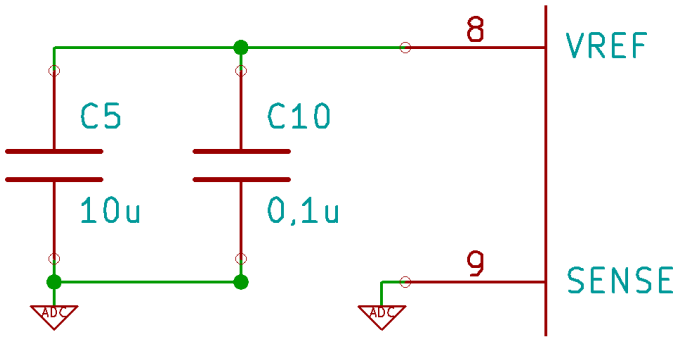

Fig. 2.30: pin VREF, pin SENSE

In questa configurazione lo switch, del morsetto meno dell'amplificatore, è collegato al partitore resistivo interno (vedi Fig. 2.31), in modo che il pin VREF sia settato a 1V e lo SPAN a 2V (= 2 x VREF).

Per completezza, anche se non vengono utilizzate nel progetto, vengono spiegate le altre possibili configurazioni. Connettendo il pin SENSE al pin VREF, e lo switch del comparatore al pin SENSE (completando il loop) si ottiene un riferimento di tensione pari a 0,5V. Se invece viene connesso un partitore resistivo esterno, come Fig. 2.32, e lo switch è connesso al pin SENSE, l'amplificatore è posto in configurazione non invertente, per cui l'uscita VREF è definita in 2.26. E risulta programmabile a seconda della scelta delle resistenze che compongono il partitore.

$$V_{REF} = 0,5 \cdot (1 + \frac{R_2}{R_1}) \tag{2.26}$$

Nella scheda Terasic_THDB_ADA i pin REFT e REFB dei canali A e B sono collegati ad un circuito capacitivo come in Fig. 2.33.

In tutte le configurazioni i pin REFT e REFB stabiliscono lo SPAN di ingresso dell'ADC. In particolare il range di ingresso dell'ADC è dato dalla 2.27.

$$SPAN = 2 \cdot (REFT - REFB) \tag{2.27}$$

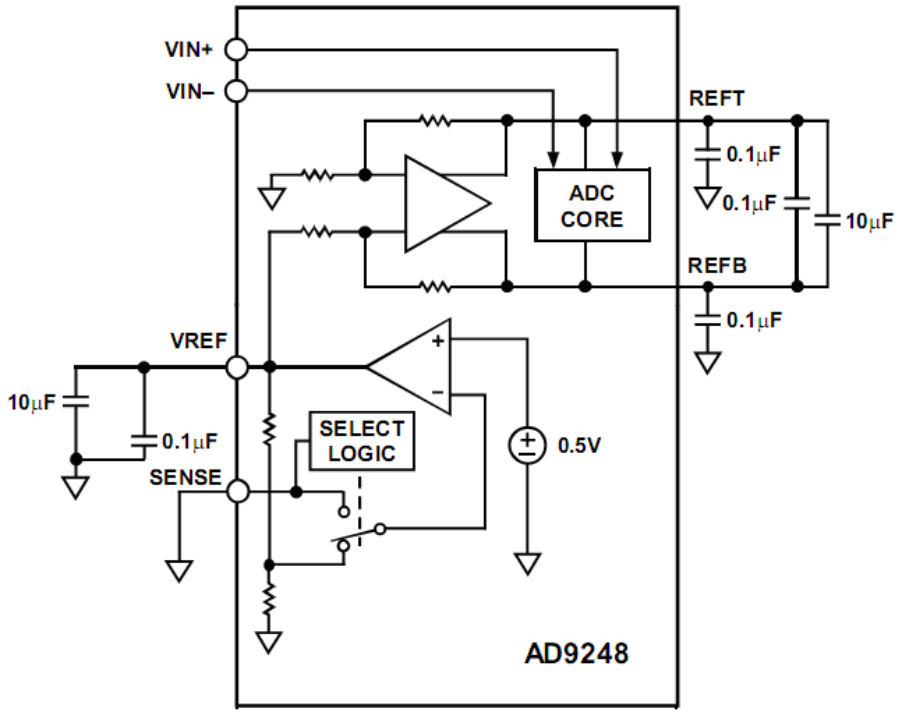

Fig. 2.31: configurazione del riferimento interno di tensione (V_{REF}) (con partitore interno)

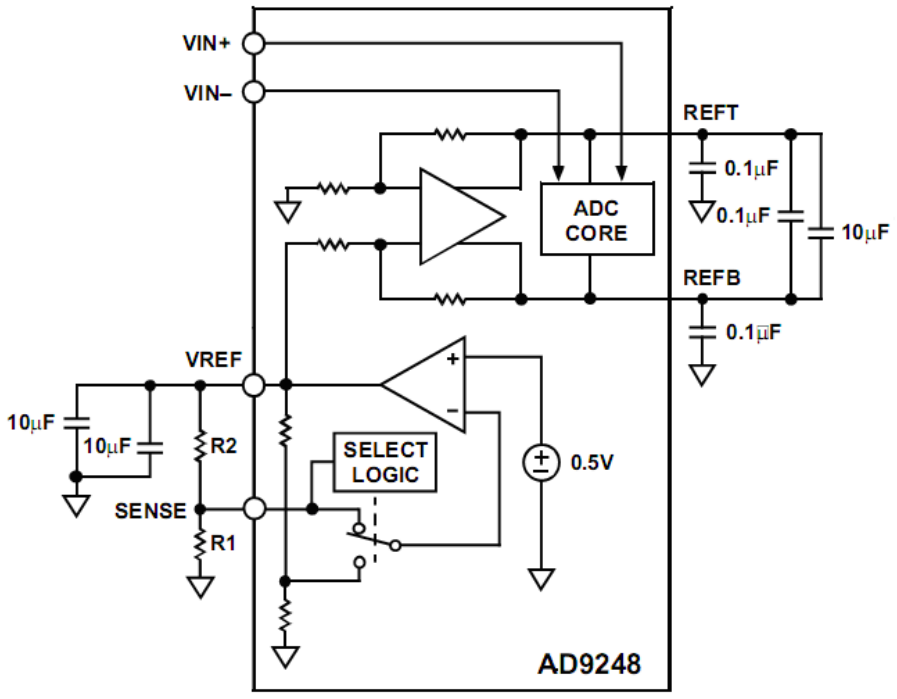

Fig. 2.32: configurazione del riferimento interno di tensione (V_{REF}) (con partitore esterno)

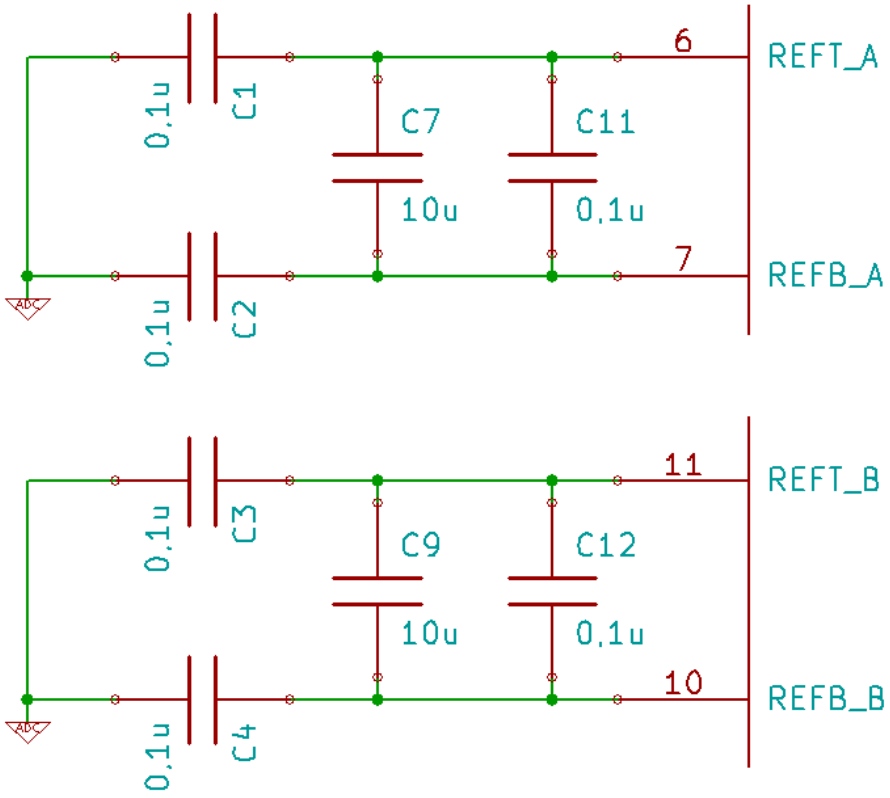

Fig. 2.33: pin REFT e REFB dei canali A e B

Le tensioni concatenate, scalate e rese differenziali, uscenti dalla scheda di interfaccia, vengono collegate all'ingresso degli ADCs attraverso la rete RC riportata in Fig. 2.34.

Si riportano infine alcuni ragionamenti sulla temporizzazione dei dati digitali a 14bit in uscita dai convertitori. L'AD9248 fornisce i dati digitali in uscita con un ritardo (dovuto alla pipeline) di 7 cicli di clock. Inoltre i dati in uscita sono disponibili dopo un ulteriore ritardo di propagazione (t_{PD} = time propagation delay) dopo il fronte di salita del segnale di clock. Tale situazione è illustrata graficamente in Fig. 2.35.

Quindi il programma caricato nell'FPGA fornirà la stessa frequenza di clock ai convertitori AD e agli accumulatori. In più gli accumulatori dovranno iniziare a processare i dati in ingresso con un ritardo, impostato sul caso peggiore, pari a 6ns; in modo da rimanere sincronizzati con i dati in uscita dai due ADCs. Per vedere l'implementazione del ritardo si rimanda alla sottosezione 3.5.1.

2.4.3 Il convertitore DA

I dati finali, elaborati dal programma caricato nell'FPGA, devono poter essere visualizzati sull'oscilloscopio per opportune misurazioni e verifiche sperimentali (vedi Fig. 2.1). Quindi si deve effettuare una trasformazione digitale/analogica. Per far questo, si sfrutta la parte della scheda Terasic_THDB_ADA dedicata a tale scopo; in particolare il convertitore AD9767.

Il convertitore AD9767 è dotato delle seguenti caratteristiche principali:

1. due DAC con profondità di 14 bits
2. velocità di campionamento fino a 125MSPS

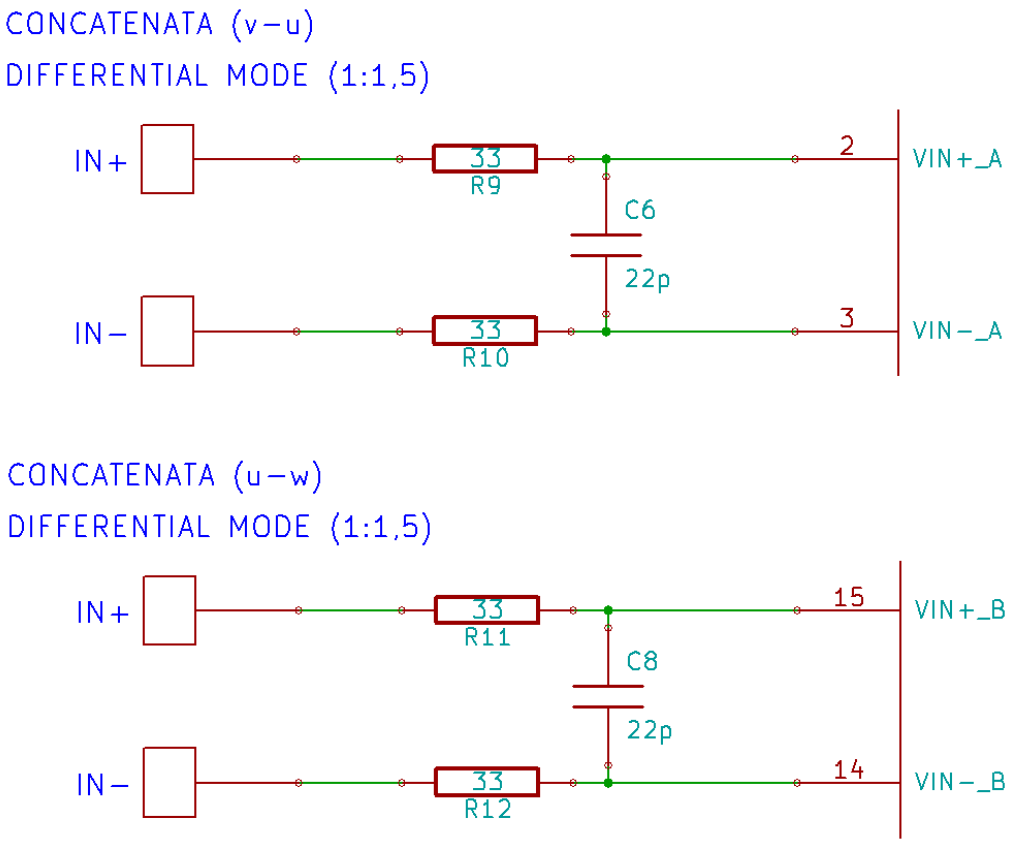

Fig. 2.34: Ingressi Differenziali

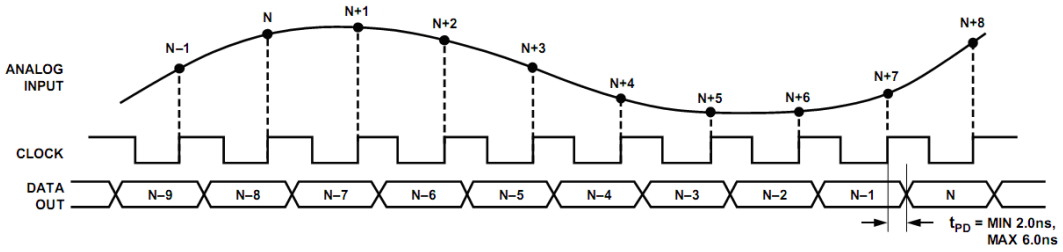

Fig. 2.35: Ritardo di propagazione dei dati di uscita

3. capacità di operare a 5V o a 3,3V

In figura Fig. 2.36 si riporta il diagramma funzionale del convertitore AD9767 compreso di pin-out.

Nel seguito del capitolo si analizzerà il funzionamento del convertitore AD9767, per ulteriori dettagli si rimanda alla [8]. In Fig. 2.37 si riporta lo schema circuitale dell'AD9767 e dell'interfacciamento con la scheda Terasic_THDB_ADA. Lo schematico del convertitore AD9767 insieme al circuito di comando, è stato estrapolato e modificato, a seconda delle esigenze di progetto, dalla [7] e disegnato con il software Kicad.

L'AD9767 è formato da due DACs, ognuno con la sua logica di controllo e le sue correnti di uscita. Ciascun DAC contiene un array PMOS in grado di fornire fino a 20mA di corrente di fondo scala (IOUTFS). L'array è diviso in 31 correnti uguali che compongono

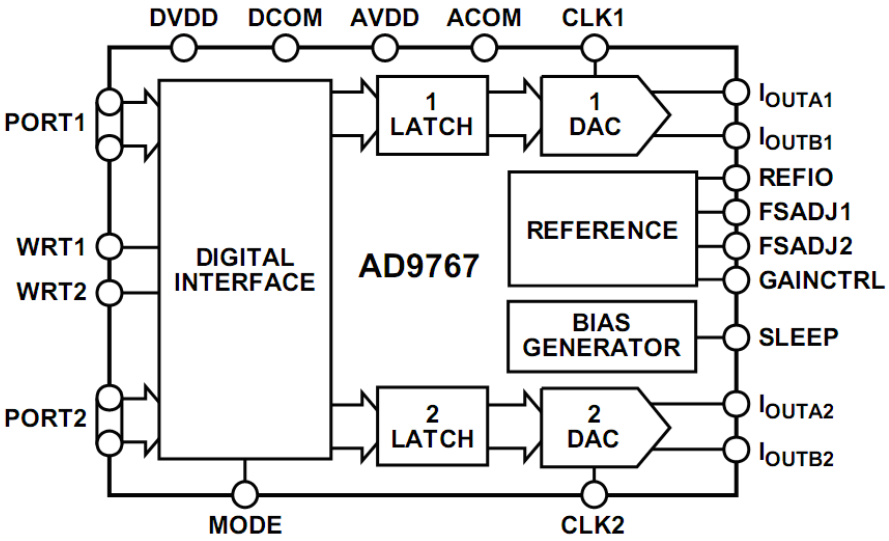

Fig. 2.36: AD9767, diagramma funzionale

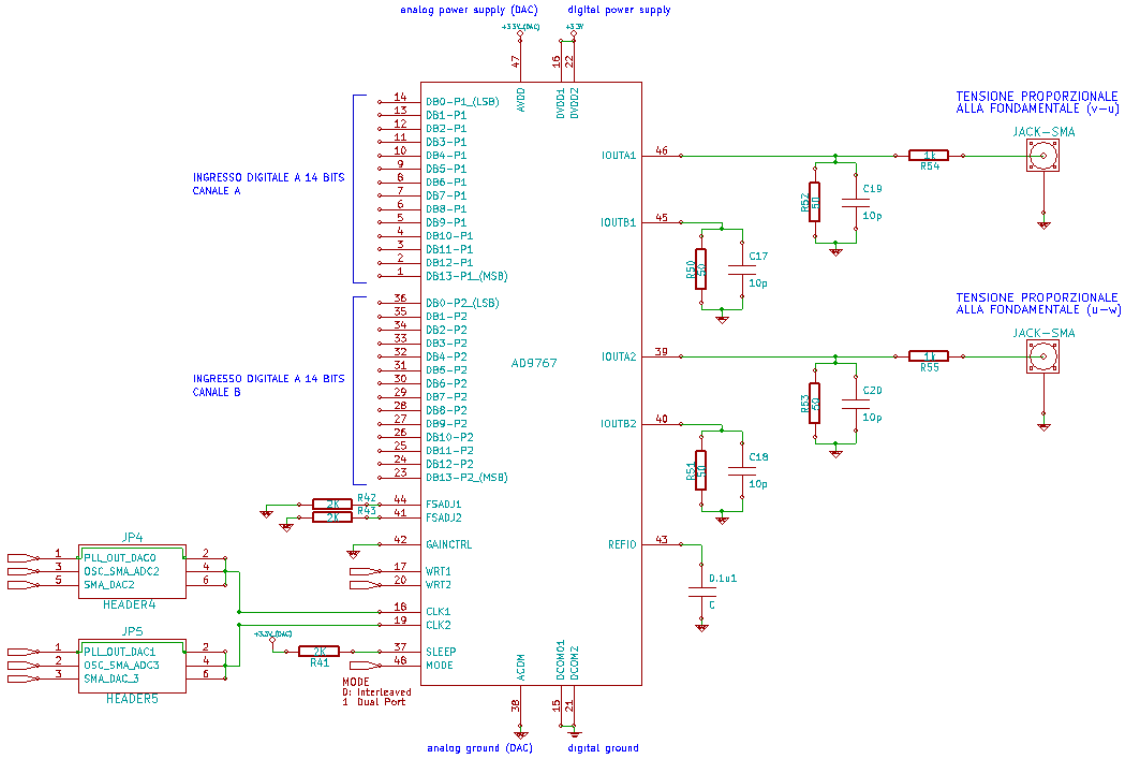

Fig. 2.37: AD9767, schema circuitale

i cinque bits più significativi (MSBs). I successivi 4 bits (o bits di mezzo) sono formati da quindici correnti uguali, il cui valore è 1/16 del valore di una delle correnti MSBs I rimanenti LSBs sono formati da un ulteriore frazionamento delle correnti dei bit di mezzo. Tutte queste correnti sono indirizzate verso uno dei due nodi di uscita di ciascun DAC (IOUTA or IOUTB) attraverso uno switch di corrente differenziale PMOS. Lo schema a blocchi è riportato in Fig. 2.38.

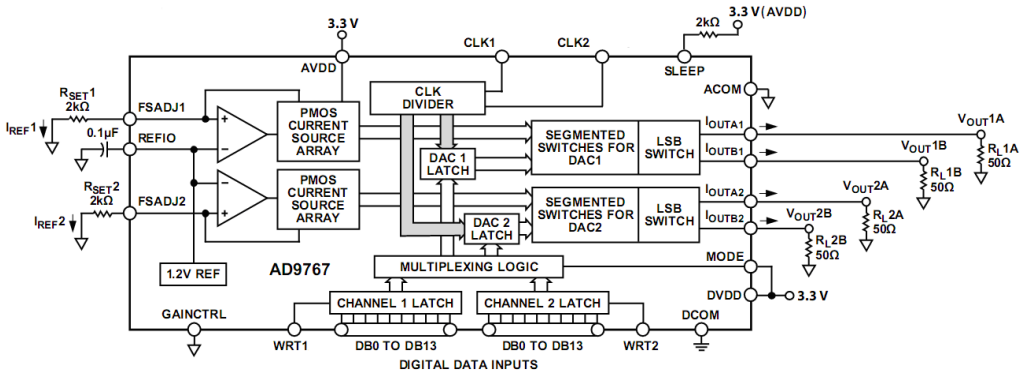

Fig. 2.38: AD9767, schema a blocchi

La parte analogica e quella digitale hanno due alimentazioni separate (AVDD e DVDD) che possono lavorare indipendentemente a 3,3 V o 5,0 V (nella scheda Terasic_THDB_ADA lavorano entrambe a 3,3V) La parte digitale è in grado di operare a 125MSPS (Mega Sample Per Second) e consiste in latches edge triggered e in circuiti logici di decodifica. La parte analogica include le fonti di correnti PMOS, gli switch differenziali associati, un riferimento di tensione pari a 1,2V, e due amplificatori per il controllo del riferimento di tensione.

L'AD9767 contiene un riferimento interno di tensione pari a 1,2V. Questo può essere semplicemente sostituito da un riferimento esterno senza peggiorare le performance. Il pin di riferimento REFIO è sia di ingresso che di uscita, a seconda se viene utilizzato il riferimento interno o un riferimento esterno di tensione. Per utilizzare il riferimento interno basta semplicemente disaccoppiare il pin REFIO dal pin ACOM (Analog Common) con una capacità da $0.1\mu F$, come illustrato nell'ingrendimento di Fig. 2.39.

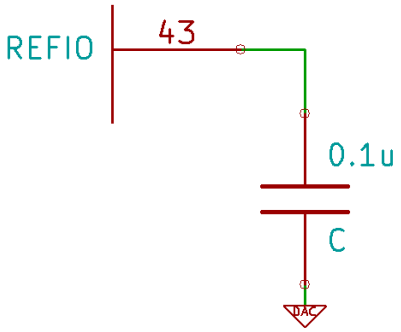

Fig. 2.39: pin REFIO

In questo modo il riferimento interno di tensione è presente al pin REFIO (questa è la configurazione adottata nella scheda Terasic_THDB_ADA). Si mostra in Fig. 2.40 lo schema a blocchi di configurazione per l'utilizzo del riferimento interno di tensione. Se la tensione al pin REFIO è utilizzata in altre parti del circuito, viene posto un buffer amplificatore esterno con una corrente di polarizzazione di ingresso inferiore ai 100nA. (Nel

progetto tale tensione non viene utilizzata da nessun'altra parte, quindi nello schematico della scheda il buffer non è presente)

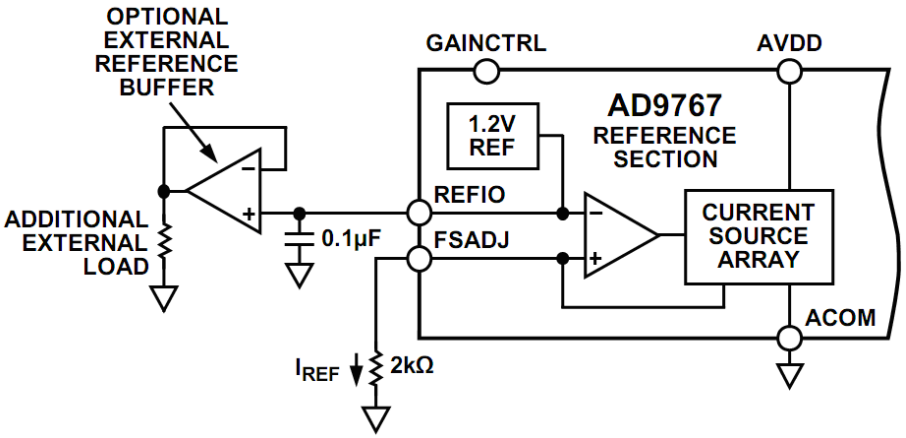

Fig. 2.40: Configurazione del riferimento interno di tensione

Anche se nel progetto tale configurazione non viene utilizzata, per completezza viene riportato in Fig. 2.41 come un riferimento esterno possa essere applicato al pin REFIO.

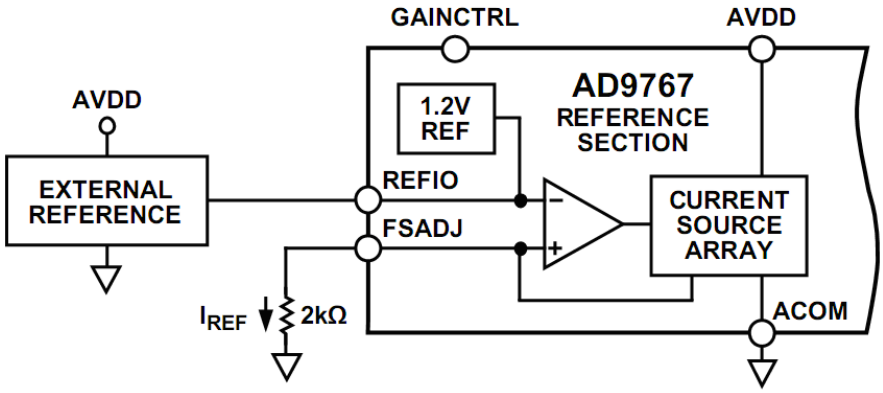

Fig. 2.41: Configurazione del riferimento esterno di tensione

Il riferimento esterno può fornire sia una tensione fissa, per aumentare l'accuratezza e le performance, sia una tensione variabile per il controllo del guadagno. La capacità di compensazione da $0.1\mu F$ non è richiesta perché il riferimento interno viene sovrascritto; e la relativa alta impedenza di ingresso al pin REFIO minimizza un eventuale effetto di carico del riferimento esterno.

L'AD9767 fornisce due guadagni indipendenti ai due DACs, ciò viene implementato fisicamente connettendo due resistenze R_{SET} distinte ai due pin FSADJ1 e FSADJ2 (come fatto nella scheda Terasic_THDB_ADA).

Nello schematico Kicad le resistenze R_{SET} corrispondono alle resistenze R42 e R43 riportate nell'ingrandimento di Fig. 2.42.

Per ridurre i costi può essere utilizzata una singola resistenza R_{SET} per impostare il guadagno di entrambi i canali.

Il pin adibito alla selezione della modalità di funzionamento è il pin GAINCTRL (Gain Control). Quando il pin GAINCTRL si trova al valore logico basso (cioè è connesso ad AGND → Analog GND), come in Fig. 2.43, viene attivato il gain control mode con due resistenze indipendenti (come nella scheda Terasic_THDB_ADA)

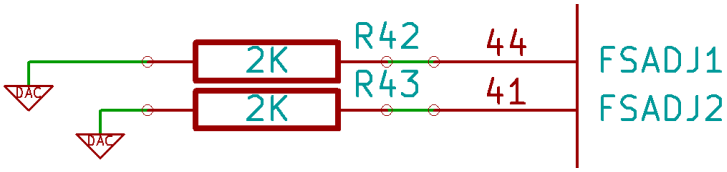

Fig. 2.42: pin FSADJ1, pin FSADJ2

Fig. 2.43: pin GAINCTRL

In questo modo le due resistenze indipendenti R_{SET} sono connesse ai pin FSADJ1 e FSADJ2 (come in Fig. 2.42).

Se il pin GAINCTRL è al valore logico alto (cioè è connesso ad AVDD → Analog VDD) viene attivato il gain control mode con una sola resistenza. In questo modo una resistenza R_{SET} è connessa al pin FSADJ1 e la resistenza connessa al pin FSADJ2 deve essere rimossa (questo modo di funzionamento non è implementato nella Terasic_THDB_ADA).

Entrambi i DACs dell'AD9797 contengono un amplificatore di controllo utilizzato per regolare il fondo scala della corrente di uscita (I_{OUTFS}). L'amplificatore di controllo è configurato come un convertitore Tensione-Corrente come mostrato in Fig. 2.40, in modo che la sua corrente di uscita (I_{REF}) sia determinata dal rapporto tra la tensione di riferimento V_{REFIO} e la resistenza esterna R_{SET} come evidenziato nella 2.28.

$$I_{REF} = \frac{V_{REFIO}}{R_{SET}} \tag{2.28}$$

La corrente I_{REF} viene riportata, col dovuto fattore di scala, alle varie fonti di correnti (della conversione DA) per settare il fondo scala della corrente di uscita I_{OUTFS}, come descritto nella 2.29.

$$I_{OUTFS} = 32 \cdot I_{REF} \tag{2.29}$$

L'amplificatore di controllo permette uno span di regolazione del fondo scala della corrente di uscita I_{OUTFS} da 1,92mA a 19,2mA, settando la corrente I_{REF} da 0,06mA a 0,6 mA, utilizzando una resistenza R_{SET} da $20k\Omega$ a $2k\Omega$. L'ampio range di controllo della corrente di uscita I_{OUTFS} fornisce alcuni benefici, tra i quali, i 20dB di regolazione forniti dalla scheda, molto utili in sistemi che richiedono il controllo del guadagno, e la possibilità di settare la potenza dissipata, che è proporzionale alla I_{OUTFS}.

Nella scheda Terasic_THDB_ADA sono state collegate ai pin FSADJ1 e FSADJ2 due resistenze da $2k\Omega$ (vedi Fig. 2.42). Pertanto la corrente di riferimento I_{REF} e la corrente di fondo scala I_{OUTFS} sono date dalle 2.30 e 2.31.

$$I_{REF} = \frac{V_{REFIO}}{R_{SET}} = \frac{1,2V}{2k\Omega} = 600\mu A \tag{2.30}$$

$$I_{OUTFS} = 32 \cdot I_{REF} = 32 \cdot 600\mu A = 19,2mA \tag{2.31}$$

Si chiarisce di seguito la logica utilizzata dai due DACs per il calcolo delle correnti di uscita. Entrambi i DACs nell'AD9767 forniscono correnti di uscita complementari, I_{OUTA}

e I_{OUTB}. I_{OUTA} fornisce una corrente vicina al valore di fondo scala (I_{OUTFS}) quando tutti i bits del segnale digitale in ingresso sono pari a uno, come evidenziato nella 2.32.

$$DAC\_CODE = 2^{14} - 1 = 16383 = 11111111111111 \tag{2.32}$$

Mentre I_{OUTB}, l'uscita complementare, fornisce corrente nulla. Le correnti I_{OUTA} e I_{OUTB} sono in funzione del codice digitale in ingresso e della corrente di fondo scala I_{OUTFS}, e possono essere espresse dalle equazioni 2.33 e 2.34.

$$I_{OUTA} = \frac{DAC\_CODE}{16384} I_{OUTFS} \tag{2.33}$$

$$I_{OUTB} = \frac{16383 - DAC\_CODE}{16384} I_{OUTFS} \tag{2.34}$$

Le due correnti di uscita tipicamente comandano una resistenza di carico direttamente o attraverso un trasformatore. Nel progetto è richiesto un accoppiamento DC, pertanto I_{OUTA} e I_{OUTB} sono direttamente connesse ad un carico resistivo R_{LOAD}, che all'altro capo è collegato alla massa analogica. Nello schematico kicad, di cui, in Fig. 2.44 si riporta un ingrandimento. Le resistenze di carico R_{LOAD} sono rappresentate dalle resistenze R50, R51, R52, R53, del valore di 50Ω.

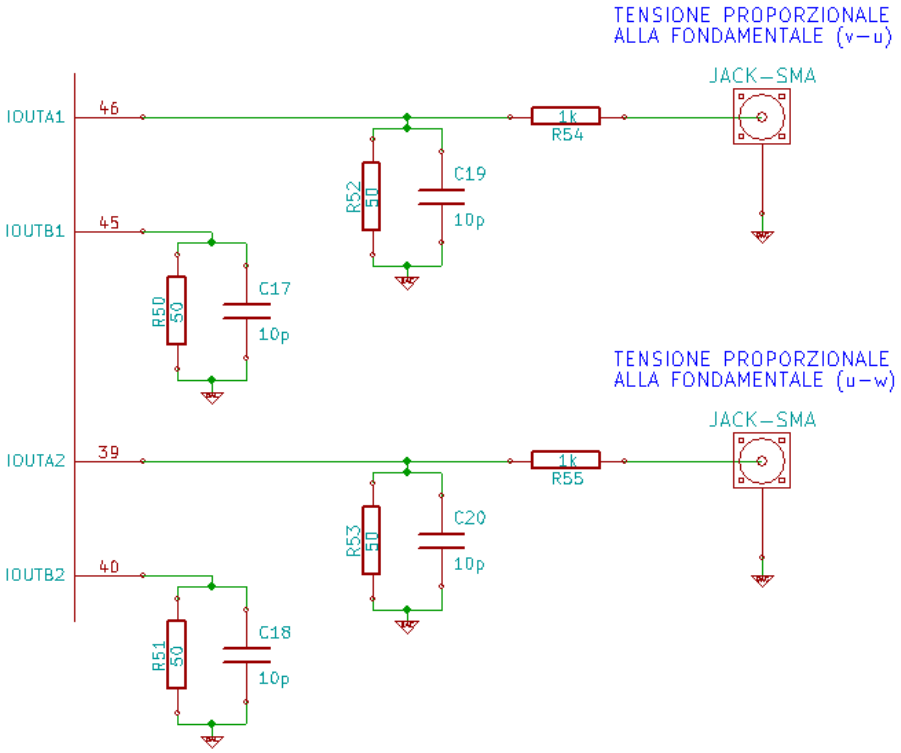

Fig. 2.44: Uscite analogiche

Le tensioni single-ended di uscita su I_{OUTA} e I_{OUTB} sono semplicemente date dalle 2.35 e 2.36.

$$V_{OUTA} = I_{OUTA} \cdot R_{LOAD} \tag{2.35}$$

$$V_{OUTB} = I_{OUTB} \cdot R_{LOAD} \tag{2.36}$$

Nel progetto viene utilizzata solamente l'uscita single-ended V_{OUTA} per entrambi i DACs. In Fig. 2.45 viene illustrato uno schema rappresentante l'uscita di uno dei due convertitori DA (per l'altro le considerazioni sono duali). La configurazione adottata in laboratorio permette di fornire in uscita un range unipolare compreso tra 0V e 960mV. Questo perché, come spiegato in questa sottosezione, la corrente di uscita (I_{OUTA}), con valore di fondo scala pari a I_{OUTFS} = 19,2mA, scorre attraverso una resistenza da 50Ω. L'uscita non utilizzata (I_{OUTB}) è connessa alla massa analogica attraverso una resistenza da 50Ω. L'oscilloscopio, per la misurazione della tensione di uscita, viene collegato al sistema attraverso un jack sma (SubMiniature version A). Il collegamento viene fatto in serie ad una resistenza da $1k\Omega$ (vedi Fig. 2.44 e Fig. 2.45). Tale resistenza non influenza la misura, in quanto l'oscilloscopio ha un'alta impedenza interna (pari a circa $1M\Omega$) e quindi il ramo composto dalla serie delle due resistenze ($1M\Omega$+$1k\Omega$) non assorbe corrente, pertanto la resistenza da $1k\Omega$ non perturba la misurazione.

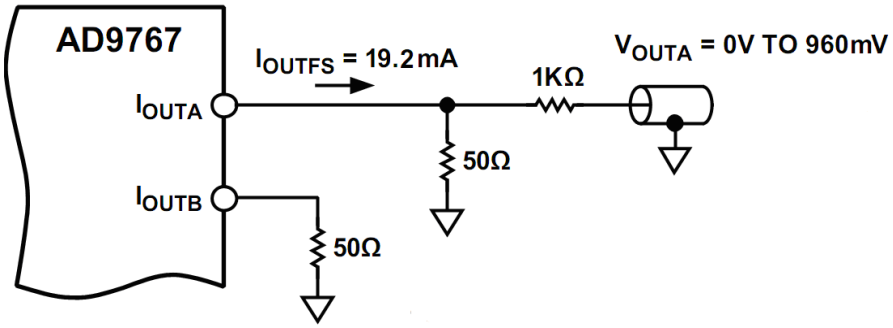

Fig. 2.45: Tensione analogica d'uscita

Pertanto tensioni nulle in ingresso al sistema corrisponderanno ad una tensione di uscita pari a 480mV. Tensioni positive in ingresso al sistema corrisponderanno a tensioni di uscita comprese tra 480mV e 960mV. Tensioni negative in ingresso al sistema corrisponderanno a tensioni d'uscita comprese tra 0V e 480mV.

Il convertitore DA processa ingressi digitali a 14bit, e la sua uscita ha un valore di fondo scala pari a 960mV; quindi la sensibilità, espressa in μV, del valore di uscita è data dalla 2.37.

$$Sensibilità\_d'Uscita = \frac{960mV}{16384} = 58,594\mu V \tag{2.37}$$

Il pin MODE, nella scheda Terasic_THDB_ADA, viene settato via software al valore logico alto (vedi sottosezione 3.5.3). Il valore alto di tensione viene riportato dall'FPGA al pin MODE attraverso il connettore HSMC (vedi capitolo 2.5). In tale condizione l'AD9767 opera in modalità ''dual-port''. Ovvero l'AD9767 funziona con due DACs distinti, quindi ogni DAC ha i suoi ingressi digitali indipendenti e le sue indipendenti linee di controllo. Se il pin MODE si trovasse al valore logico basso, l'AD9767 funzionerebbe in modalita 'interleaved'. Tale modalità non viene utilizzata nella scheda Terasic_THDB_ADA. Per eventuali approfondimenti si rimanda alla [8]. I dati digitali entrano nel dispositivo attraverso i due latches di ingresso del canale 1 e del canale 2, come riportato nello schema a blocchi di Fig. 2.46. Tali dati vengono poi trasferiti rispettivamente al latch del DAC1 e al latch del DAC2. Una volta che i dati sono caricati nei latches dei due DACs, le corrispondenti uscite analogiche si portano ai nuovi valori (che sono le conversioni analogiche dei dati digitali salvati nei latches dei due DACs).

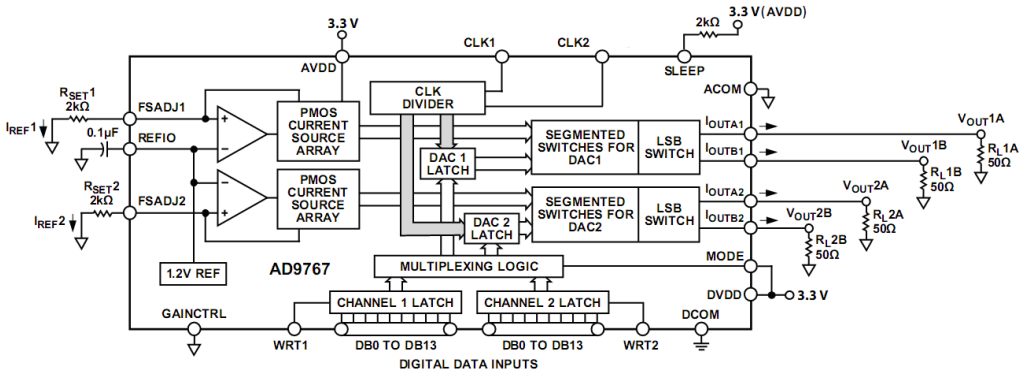

Fig. 2.46: AD9767, schema a blocchi

Il pin WRT1 e il pin WRT2, sono dei pin che forniscono il clock di controllo rispettivamente al latch del canale 1 di ingresso e al latch del canale 2 di ingresso. Il pin CLK1 e il pin CLK2, sono dei pin che forniscono il clock di controllo rispettivamente al latch del DAC1 e al latch del DAC2. I segnali di set di tutti i latches sono aggiornati sul fronte di salita dei loro rispettivi segnali di controllo (WRT per i latches dei canali di ingresso e CLK per i latches dei DAC). Il fronte di salita del segnale CLK avviene prima o simultaneamente con il fronte di salita del segnale WRT. I clock dei pin CLK1, CLK2, WRT1, WRT2, nel progetto vengono dati via software; attraverso un PLL (vedi sottosezione 3.5.2) che, partendo dal contatore interno a 50MHz della FPGA, fornisce il clock desiderato. Dal punto di vista circuitale, il clock ai pin WRT1 e WRT2 viene fornito dall'FPGA attraverso il connettore HSMC (vedi capitolo 2.5) mentre, a valle del connettore HSMC, per forzare il clock dei pin CLK1 e CLK2 al valore creato via software, si deve applicare un jumper che connetta i terminali 1 e 2 dei connettori JP4 e JP5, come illustrato in Fig. 2.47.

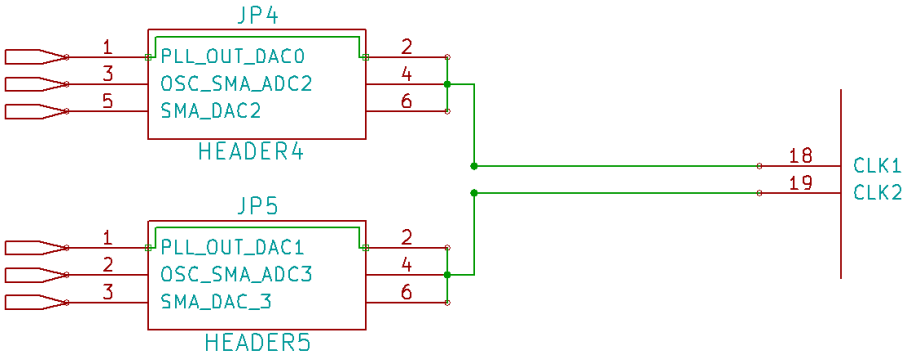

Fig. 2.47: pin CLK1, pin CLK2

I terminali collegati ai pin OSC_SMA_ADC2/3 e SMA_DAC2/3 permetterebbero di utilizzare un clock esterno alla scheda, opportunamente collegato attraverso i jack sma (SubMiniature version A) appositi. Nel progetto tale possibilità non è stata utilizzata.

Infine l'AD9767 ha una funzione di spegnimento, che azzera la corrente di uscita e riduce la corrente di alimentazione a valori inferiori a 8,5mA. Tale funzione si attiva se la tensione di alimentazione supera il valore nominale di 3,3V o se la temperatura di esercizio supera il range consentito. Questa modalità viene attivata applicando un valore logico alto al pin SLEEP. Il valore di soglia, oltre il quale il pin SLEEP legge un valore logico alto è

dato dalla 2.38.

$$V_{SOGLIA} = 0,5 \cdot AVDD = 0,5 \cdot 3,3V = 1,65V \tag{2.38}$$

Nell'ingrandimento di Fig. 2.48 si osserva la configurazione del pin SLEEP sulla scheda Terasic_THDB_ADA.

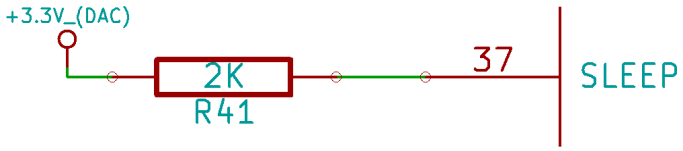

Fig. 2.48: pin SLEEP

Se un aumento della tensione di alimentazione portasse il potenziale del pin SLEEP al valore di soglia pari a 1,65V, il pin SLEEP leggerebbe tale tensione come un valore logico alto, e quindi attiverebbe la funzione di spegnimento

2.5 Scheda FPGA

Una FPGA (Field Programmable Gate Array) è un insieme di porte logiche programmabili. E' dotata di grande velocità di elaborazione, e viene utilizzata per realizzare funzioni logiche, macchine sequenziali sincrone e asincrone, reti combinatorie. E' inoltre adatta al controllo di led e display. Per lo sviluppo del progetto è stata utilizzata l'FPGA_Cyclone_III prodotta dalla Altera, e montata dalla stessa azienda sulla scheda di test Cyclone_III_FPGA_Starter_Board. L'FPGA_Cyclone_III_Starter_Board è programmabile attraverso l'ambiente di sviluppo Quartus II, rilasciato dall'Altera. Tale ambiente facilita la programmazione della scheda, fornendo agli utenti molti blocchi preprogrammati, facenti numerosissime funzioni. L'utente può comunque realizzare nuovi blocchi attraverso il linguagio di programmazione Verilog VHDL. La scheda FPGA può espandere le sue funzionalità attraverso il collegamento ad altre schede, sfruttando il connettore HSMC (High Speed Mezanine Card) Come precedentemente detto nella sottosezione 2.4.1, la scheda FPGA si collega alla scheda di acquisizione analogico/digitale attraverso il medesimo connettore. In Fig. 2.49 è riportata l'FPGA_Cyclone_III_Starter_Board con le sue caratteristiche hardware principali.

La scheda FPGA è caratterizzata dai seguenti elementi:

- FPGA Altera Cyclone III EP3C25, contenuta in un package avente 324 pin, è composta da 25000 elementi logici programmabili, 0,6Mbit di blocchi di memoria e 16 blocchi moltiplicatori.
- Connettore HSMC (High Speed Mezzanine Card) per l'interfacciamento con schede aggiuntive. Tale connettore è una versione modificata del connettore standard ad alta velocità Samtec. Le modifiche apportate forniscono maggiore immunità ai disturbi e minor distorsione del segnale trasmesso.Attraverso questo connettore vengono inoltre forniti due segnali rispettivamente a 3,3V e 12V, e i relativi collegamenti di massa, necessari ad alimentare le schede di espansione connesse. Il connettore HSMC in totale presenta 84 pin di I/O bidirezionali, ad eccezione dei pin riservati alla trasmissione dei segnali di clock. I pin dell'HSMC sono caratterizzati da un livello logico alto pari a 2,5V. L'FPGA riconosce un livello logico alto all'interno di un range di tensione compreso tra 1,7V e 4,1V, viceversa riconosce un livello logico basso all'interno di un range di tensione compreso tra -0,5V e 0,7V.

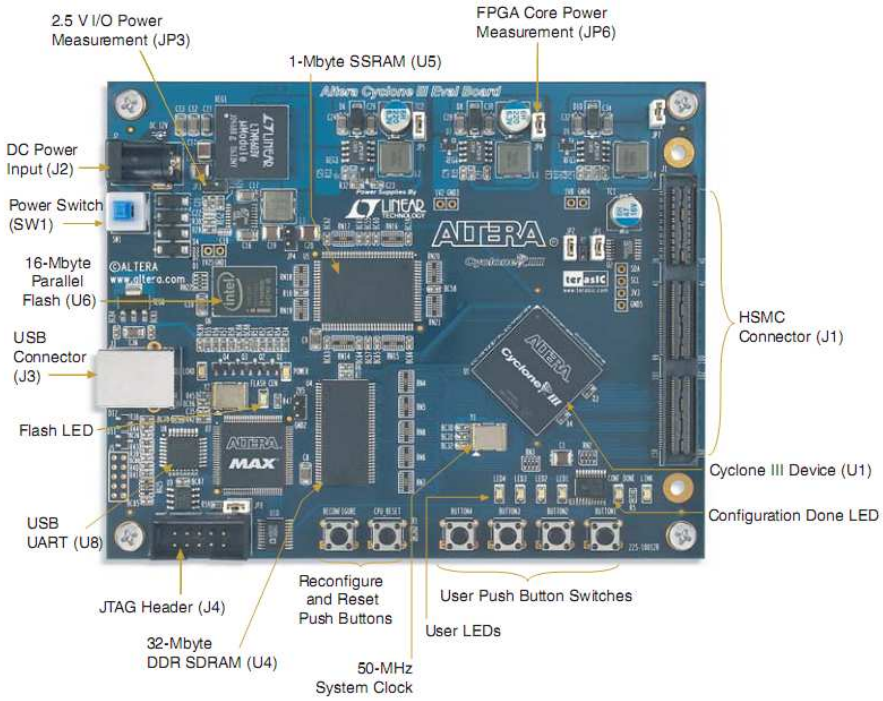

Fig. 2.49: FPGA_Cyclone_III_Starter_Board

- Connettore USB per l'interfacciamento con il PC.
- Memoria DDR SDRAM da 32Mbyte.
- Memoria Flash da 16Mbyte per la configurazione dell'FPGA e il salvataggio dei dati allo spegnimento.
- Memoria SSRAM ad alta velocità da 1 Mbyte.
- 4 pulsanti programmabili, che, se premuti, forniscono un valore logico alto, finché non vengono rilasciati. 2 pulsanti non programmabili: il pulsante System-reset e il pulsante User-reset. Rispettivamente il primo serve per la riconfigurazione dell'FPGA attraverso la memoria flash; il secondo serve per resettare i progetti caricati nel dispositivo.
- 4 led programmabili. 3 led non programmabili: Power Led, Configuration Led, Flash Signal Led. Rispettivamente l'accensione del primo indica che la scheda è alimentata, l'illuminazione del secondo indica una programmazione dell'FPGA avvenuta con successo, l'attivazione del terzo indica un accesso alla memoria Flash.
- Circuito per la generazione del clock interno alla scheda, che fornisce un segnale a 50Mhz. Tutti gli altri segnali di clock necessari vengono ricavati da esso attraverso dei PLL (anelli ad aggancio di fase) Tali dispositivi forniscono i clock alla memoria flash, alla memoria SSRAM, al connettore HSMC (e di conseguenza alle schede collegate) E' inoltre possibile programmare a piacere i PLL in modo da fornire, a seconda delle esigenze, delle particolari frequenze di clock ai pin di I/O

La figura Fig. 2.49 è stata estrapolata dalla [9], alla quale si rimanda per ulteriori approfondimenti.

Nel progetto si è avuto l'esigenza di collegare il sincronismo uscente dalla scheda di interfaccia (vedi sottosezione 2.3.2) all'FPGA. Per far questo si è utilizzato un pin sul connettore HSMC libero. In particolare si è utilizzato il pin bidirezionale HSMC_RX_p8 denominato all'interno dell'FPGA con il nome pin P2.

Capitolo 3

SOFTWARE

3.1 Introduzione

Per la realizzazione del programma da caricare nell'FPGA è stato utilizzato l'ambiente di sviluppo `Altera Quartus II Web Edition versione 9.1 Service Pack 1`, scaricabile gratuitamente dal sito dell'azienda riportato in [10]. Inoltre in [11] è possibile scaricare il manuale. Tale software di programmazione contiene una notevole libreria di blocchi pre-programmati facenti molte funzioni. In questo progetto tale funzionalità è stata largamente utilizzata.

A seconda delle esigenze, sono stati inoltre creati dei blocchi, utilizzando il linguaggio di programmazione Verilog HDL. In internet si possono trovare molti manuali, tra i quali in [12] ne viene segnalato uno. Tutte le videate riportate nelle figure di questo capitolo sono state estrapolate dal `Quartus II`.

3.2 Installazione del software

Prima di effettuare l'installazione con il DVD fornito insieme alla scheda FPGA, si consiglia di verificare la presenza di versioni più recenti sul sito dell'Altera, [10]. Nel progetto è stata utilizzata, come indicato sopra, la versione 9.1. Ora è già presente la versione 10. Si sottilinea il fatto che da qui in poi ci si riferirà alla versione utilizzata nel progetto (e non all'ultima release).

L'installazione è molto semplice; una volta aperto il file di setup comparirà la schermata riportata in Fig. 3.1. Poi basterà proseguire nelle varie schermate, accettando la licenza di utilizzo ed infine scegliendo la cartella di destinazione.

Ultimata l'installazione è possibile avviare il programma. La prima schermata chiede se si desidera acquistare una licenza d'uso 'professionale' o iniziare ad utilizzare la licenza gratuita Web Edition. Basterà quindi scegliere la seconda opzione. Tale tipo di licenza è stata ampiamente sufficiente a soddisfare tutte le esigenze di implementazione richieste dal progetto. In Fig. 3.2 viene riportata la schermata d'avvio del programma; la quale fornisce tre scelte principali. Inizialmente si consiglia di utilizzare il link relativo ai vari tutorial, molto utili al primo approccio col programma. Oppure, se è già stato creato un progetto, si può scegliere di aprirlo, cliccando nella casella *Open Existing Project*. Per iniziare da zero occorre spuntare la casella *New Project Wizard* che rimanda alla procedura guidata per la creazione di un nuovo progetto, approfondita nella sezione seguente.

3.3 Creazione di un nuovo progetto

Una volta spuntata la voce *New Project Wizard* compare una finestra nella quale bisogna inserire nell'ordine: il nome della cartella che conterrà il nuovo progetto, il nome del progetto, e il nome della *top-level design entity* (che deve essere lo stesso del progetto). La *top-level design entity* è l'entità principale del progetto stesso. Nella schermata successiva viene chiesto se si desidera includere file specifici all'interno del progetto. Immaginando di partire da zero si può procedere alla schermata successiva che viene riportata in Fig. 3.3. Nella quale si deve selezionare l'FPGA in possesso. Nell'FPGA_Cyclone_III_Starer_Board in laboratorio è montata una `Cyclone III EP3C25F324C8`.

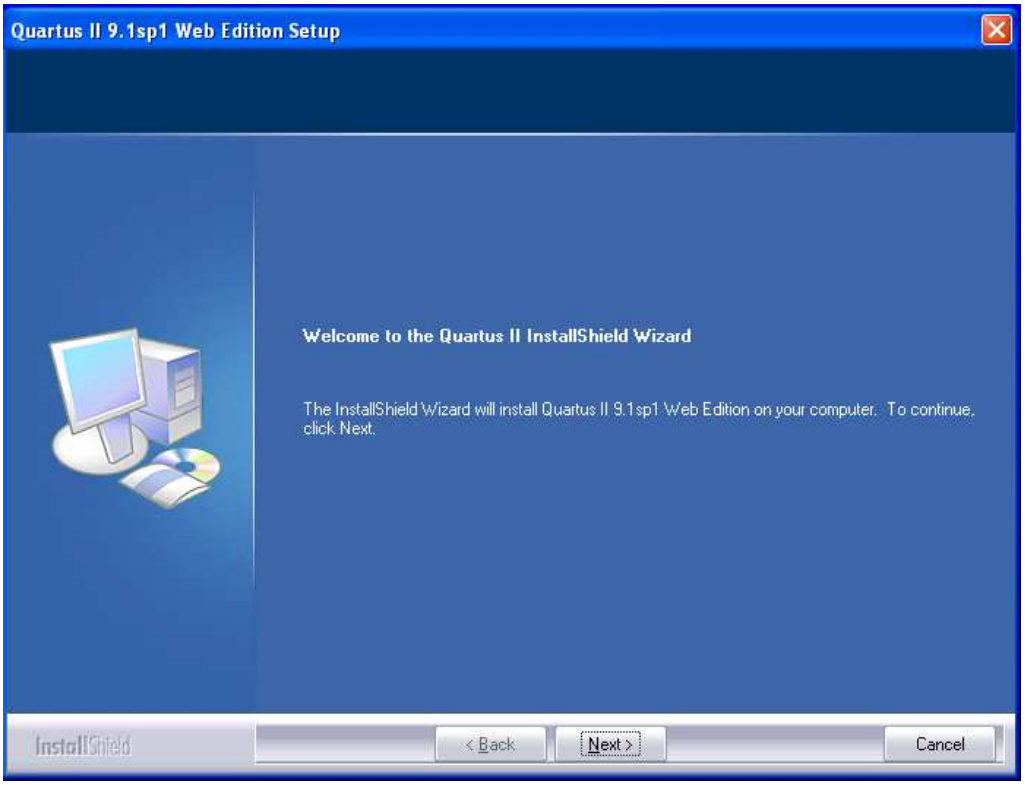

Fig. 3.1: Quartus II: Schermata di installazione

Fig. 3.2: Quartus II: Schermata iniziale

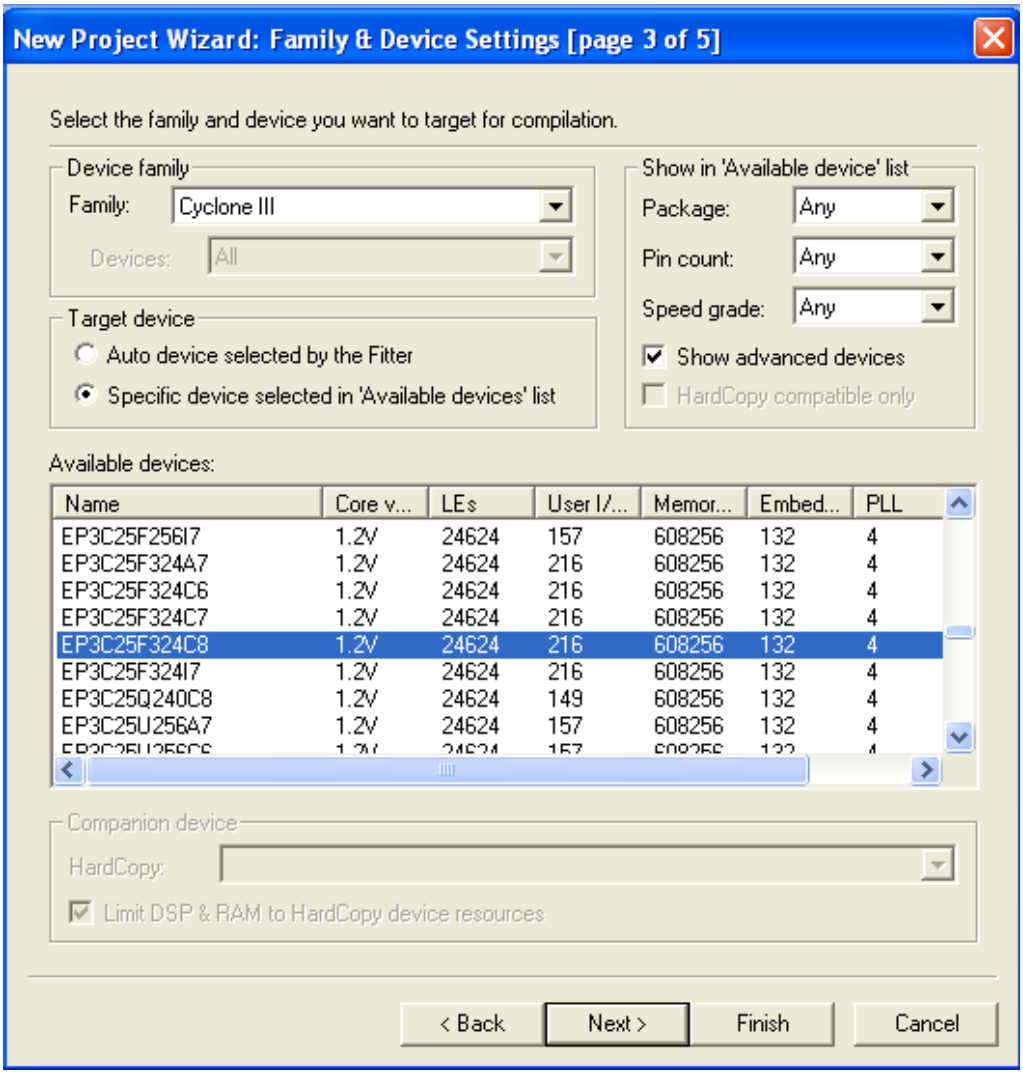

Fig. 3.3: Finestra di selezione dell'FPGA

La schermata successiva può essere velocemente saltata, in quanto non si ha l'esigenza di inserire particolari tools di simulazione e analisi temporale, al di fuori del software a disposizione. Nell'ultima schermata riepilogativa basta cliccare il pulsante *Finish*, in questo modo viene creata la cartella del progetto. All'interno della quale sono presenti i due files seguenti:

- *nome_progetto.qpf*
- *nome_progetto.qsf*

Si fa notare come il file con estensione *.qpf* sia il file di apertura del programma. Ossia per riutilizzare il progetto una volta chiuso, basterà un doppio clic sul file *nome_progetto.qpf*.

Il passo successivo consiste nella creazione della parte centrale del progetto, vale a dire un nuovo schematico di lavoro. Per far questo selezionare nei menù a tendina il percorso *File* → *New* e nella finestra che compare scegliere la voce *Block Diagram/Schematic File*. Successivamente salvare il nuovo schematico selezionando *File* → *Save As*, il nome dello schematico deve essere lo stesso del progetto. In questo modo verrà creato nella cartella del progetto un nuovo file: *nome_progetto.bdf*. Il file *nome_progetto.bdf* è il cuore del progetto. Sul quale avviene la programmazione vera e propria. Infatti è su di esso che

verrà implementato l'Algoritmo di calcolo della media mobile, che verrà spiegato nella sezione successiva.

3.4 Algoritmo di calcolo della media mobile

3.4.1 Estrazione del valore medio

Il software caricato nell'FPGA deve essere in grado di calcolare il valore medio, in ogni periodo di commutazione della PWM, dei segnali digitali campionati in ingresso. Quindi in generale, dal punto di vista matematico deve essere in grado di implementare l'equazione 3.1

$$V_N = \frac{\sum_{i=0}^{N} v_i}{N} \tag{3.1}$$

Tale equazione, restituisce la media degli N campioni processati. Immaginando ora che in un periodo di PWM siano presenti N campioni, se l'operazione di media venisse applicata su più periodi PWM, si otterebbe la media mobile del segnale in ingresso. Tale, equazione è implementabile, a patto di fare alcuni ragionamenti sulla divisione per gli N campioni. Nell'algoritmo di controllo, è necessario, per poter visualizzare i dati in uscita effettuere un troncamento degli stessi. L'operazione di troncamento è paragonabile alla divisione per N campioni nell'operazione di media.

Ovviamente troncando i dati, non si divide effettivamente per N, ma per un numero $\geq$ N (espresso in base 2), a seconda della frequenza di campionamento scelta. Ovvero, a parità di valori entranti in ingresso, maggiore sarà la frequenza di campionamento, maggiore sarà il numero accumulato nel periodo; quindi maggiore dovrà essere il troncamento da effettuare. Nel capitolo 5 verrà approfondito, come un troncamento di un numero di campioni strettamente maggiore di N, provochi una perdita di sensibilità del sistema. Ciò non inficia assolutamente il risultato finale, infatti si vedrà come il sistema risulta sensibile ad una tensione media di ingresso inferiore a 50mV. Ottimo risultato se si considera di avere a che fare con tensioni di ingresso di centinaia di volt.

Infine sempre nel capitolo 5, si potrà osservare come la scelta di una opportuna frequenza di campionamento dei dati in ingresso, porti ad un aumento della sensibilità del sistema.

3.4.2 Descrizione generale dell'algoritmo

Si premette, come già spiegato nella descrizione hardware del sistema, che il progetto consiste nell'elaborazione di due concatenate del motore. Da qui in poi si farà riferimento ad una sola delle due. In quanto tutte le operazioni eseguite, e le considerazioni fatte per l'una, risultano identiche per l'altra.

L'algoritmo di controllo è stato riportato nello schema a blocchi di Fig. 3.4.

In ingresso al convertitore AD entra la tensione concatenata, prelevata dall'inverter, attenuata e resa differenziale, come spiegato nella sezione 2.2 e nella sottosezione 2.3.1. Il convertitore AD, operando com una frequenza pari a 20Mhz, trasforma il segnale di ingresso in un segnale digitale a 14bit in codifica Offset Binary. Il segnale digitalizzato entra nell'FPGA e viene processato dal software caricato nella scheda, come spiegato di seguito.

Il segnale digitale subisce una conversione da Offset Binary a Complemento a Due. Successivamente viene portato contemporaneamente in ingresso a due accumulatori, che operano a 20Mhz con uno sfasamento iniziale di 6ns (vedi sottosezione 2.4.2), e che sono attivi in maniera alternata, grazie all'utilizzo dei segnali `Sload_Diretto` e `Sload_Negato`. Tali segnali sono ricavati dal segnale di `Sincronismo PWM` come riportato nella Fig. 3.5.

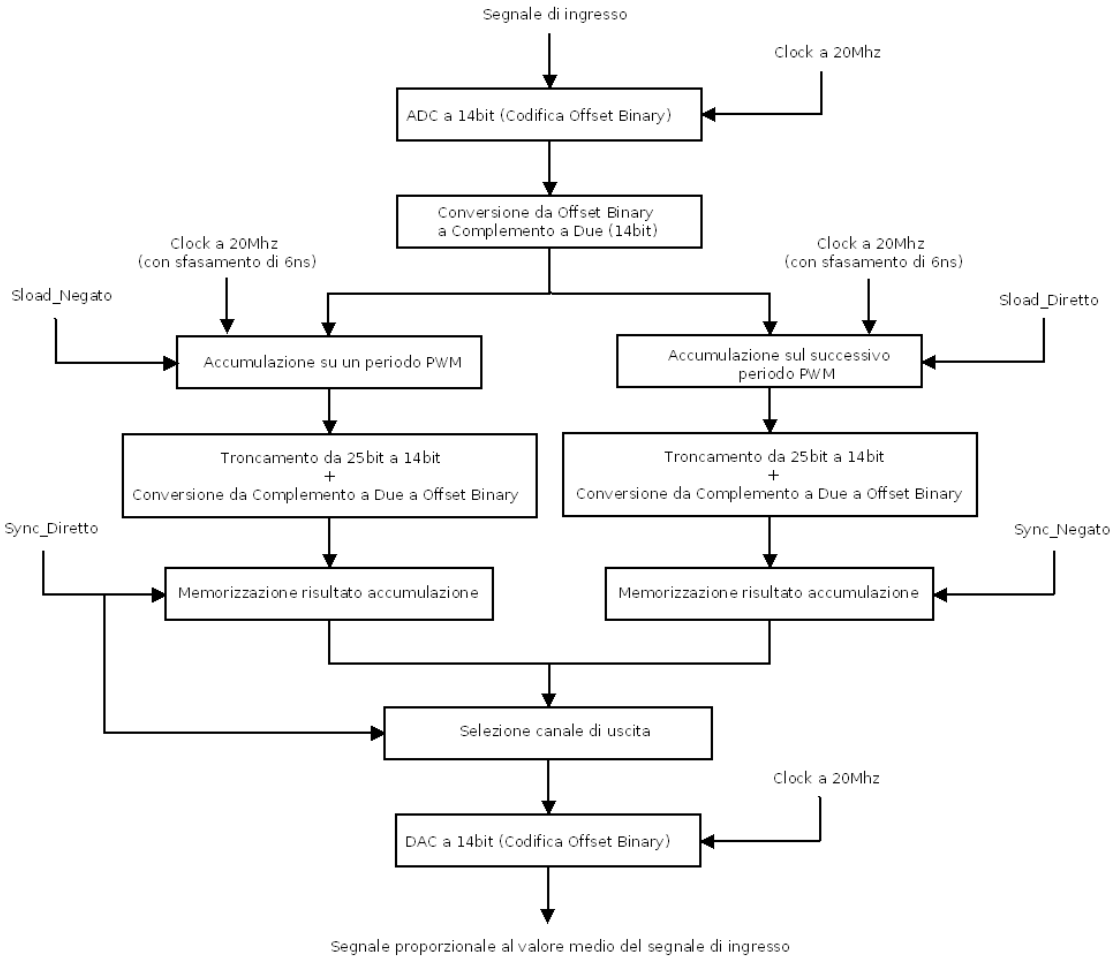

Fig. 3.4: Algoritmo di calcolo della media mobile

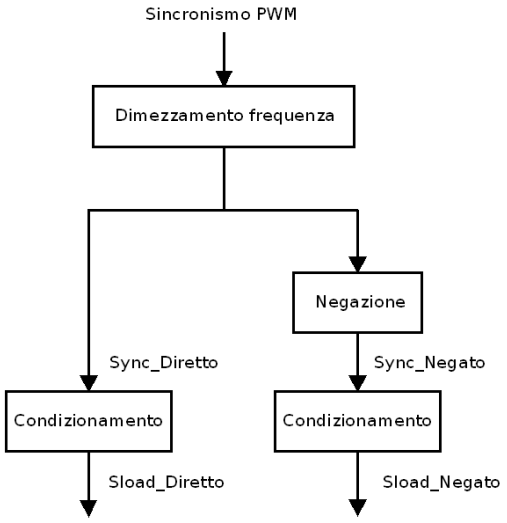

Fig. 3.5: Segnali di sincronismo

Ogni accumulatore nel periodo di PWM in cui è attivo effettua la sommatoria dei dati in ingresso a 14bit; la somma totale viene aggiornata progressivamente su un registro a 25bit, interno a ciascun accumulatore. Lo sdoppiamento del canale è stato necessario per fornire agli accumulatori il tempo di resettarsi. Ciò avviene nel periodo PWM in cui l'accumulatore dell'altro ramo sta effettuando la sommatoria.

Pochi istanti prima del reset dell'accumulatore, il dato di uscita (contenente la sommatoria dei dati in ingresso in un periodo PWM) viene riportato su 14bit e riconvertito in Offset Binary (per essere compatibile con il DAC d'uscita); infine viene memorizzato in un registro apposito. Tale registro aggiorna la sua uscita in funzione dei segnali `Sync_Diretto` e `Sync_Negato`, (vedi Fig. 3.5) a seconda del canale di riferimento, con la seguente modalità.

Quando l'accumulatore del proprio ramo è attivo l'uscita del registro è trasparente all'ingresso, pochi istanti prima del reset dell'accumulatore, il registro blocca l'uscita al valore finale della sommatoria sul periodo, uscente dall'accumulatore. Infine il blocco `Selezione canale di uscita`, seleziona in maniera alternata, il canale corretto da riportare in uscita. Questo è possibile grazie all'utilizzo del segnale `Sync_Diretto`.

A valle dell'elaborazione software il convertitore DA, operando con una frequenza di 20Mhz, effettua la conversione analogica, come descritto nella sottosezione 2.4.3.

Nel seguito del capitolo sarà spiegato in maniera più dettagliata il funzionamento generale appena accennato, inoltre verrà descritta l'implementazione software dell'algoritmo.

3.4.3 Codifica Offset Binary

Come già spiegato nella sottosezione 2.4.2 l'ADC produce dei dati d'uscita a 14bit in formato Offset Binary (OB). Tale formato ha delle corrispondenze col Complemento a Due (C2) come si può osservare nella Tab. 3.1.

Valore Decimale	Offset Binary	Complemento a Due
7	1111	0111
6	1110	0110
5	1101	0101
4	1100	0100
3	1011	0011
2	1010	0010
1	1001	0001
0	1000	0000
-1	0111	1111
-2	0110	1110
-3	0101	1101
-4	0100	1100
-5	0011	1011
-6	0010	1010
-7	0001	1001
-8	0000	1000

Tab. 3.1: Confronto codifiche (Esempio a 4bit)

In particolare entrambe le codifiche, a parità di numero di bit a disposizione, rappresentano la stessa quantita di numeri. Se viene chiamato M il numero di bit a disposizione, entrambe le codifiche possono rappresentare 2^M numeri. Lo span dei numeri rappresentabili è compreso tra $-[2^{(M-1)}]$ e $+[2^{(M-1)}] - 1$. Come si può osservare dalla Tab.

3.1 le due codifiche si differenziano solamente per il MSB (Most Significant Bit). Infatti mentre nella codifica in Complemento a Due i numeri negativi hanno un 1 come MSB e i numeri positivi hanno uno 0; per la codifica Offset Binary vale il contrario. Quindi, intuitivamente risulta facile il passaggio da una codifica all'altra. Sarà sufficiente negare il bit più significativo.

Per quanto detto, il converitore AD, (14bit) trasformerà un segnale analgico, corrispondente ad una tensione nulla, nella cosiddetta stringa di offset, pari a tredici zeri e il MSB posto a 1 (10000000000000). La sua traduzione in Complemento a Due corrisponde ad una stringa composta da 14bit tutti a zero (00000000000000).

Nel progetto risulta maggiormente utile sfruttare la codifica in Complemento a Due. In quanto, in tale formato, le operazioni di somma e sottrazione, sono entrambe eseguite soltanto utilizzando operazioni di somma. Quindi nello schematico `Quartus II` sarà sufficiente utilizzare un semplice blocco sommatore.

3.4.4 Scelta dei registri di uscita degli accumulatori

Una delle prime considerazioni da fare risulta essere il dimensionamento del registro di uscita degli accumulatori, vediamo di seguito come.

La modulazione PWM lavora ad una frquenza pari a 10Khz, mentre il convertitore AD opera a 20Mhz. Pertanto dalla 3.2 si può ottenere il numero di dati campionati dall'ADC in un periodo PWM.

$$N = \frac{20Mhz}{10Khz} = 2000\ campioni \tag{3.2}$$

Successivamente il progetto è stato dimensionato sul caso peggiore, per essere sicuri di non avere errori di overflow nelle accumulazioni. Il massimo numero rappresentabile in Offset Binary a 14bit risulta essere la stringa con 14 bit a 1; che, convertito in deciamle, corrisponde al numero $2^{14-1} - 1 = 8191$. Se ogni campione entrante negli accumulatori rappresentasse il massimo numero possibile, il registro di uscita di ciascun accumulatore dovrebbe contenere il numero decimale dato dalla 3.3.

$$8191 \cdot 2000 = 16382000 = max\ N^{\circ}\ accumulabile \tag{3.3}$$

Tale numero decimale, può essere rappresentato in binario OB o C2, con almeno 25bit a disposizione. Infatti la 3.4 restituisce un numero maggiore di quello ricavato nella 3.3.

$$2^{25-1} - 1 = 16777215 = max\ N^{\circ}\ rappresentabile\ in\ OB\ o\ C2\ con\ 25bit \tag{3.4}$$

Invece 24bit non sarebbero sufficienti, infatti la 3.5 restituisce un numero minore di quello ricavato nella 3.3.

$$2^{24-1} - 1 = 8388607 = max\ N^{\circ}\ rappresentabile\ in\ OB\ o\ C2\ con\ 24bit \tag{3.5}$$

Al fine di effettuare la somma corretta dei dati convertiti dall'ADC, e successivamente trasformati in Complemento a Due a 14bit, occorre quindi convertirli in dati a 25bit. Di seguito viene spiegata la teoria del procedimento, anche se non è stato necessario introdurre alcun blocco apposito, in quanto gli accumulatori utilizzati nello schematico `Quartus II`, implementano tale procedura.

Per estendere i bit di un numero in Complemento a Due è sufficiente copiare il MSB alla sua sinistra, fino a raggiungere il numero di bit desiderato. Nelle 3.6 e 3.7 sono riportate le estensioni dei numeri +5 (0101) e -5 (1011) da 4bit a 7bit.

$$(0101)\ C2\ a\ 4bit = (0000101)\ C2\ a\ 7bit \tag{3.6}$$

$$(1011)\ C2\ a\ 4bit = (1111011)\ C2\ a\ 7bit \tag{3.7}$$

Nella prossima sezione sarà approfondita l'implementazione software vera e propria dell'Algoritmo di calcolo della media mobile.

3.5 Implementazione dell'Algoritmo

3.5.1 Generazione dei segnali di Clock

Prima di tutto si deve introdurre il clock a 50Mhz dell'FPGA nello schematico. Per far questo, si crea un nuovo pin di input. Un pin di ingresso nello schematico non è altro che un puntatore ad un pin fisico presente nell'FPGA. Creando, con il procedimento spiegato a breve, un pin di ingresso, si chiede al programma di leggere il segnale presente sul pin fisico dell'FPGA, puntato dal pin di input dello schematico, e di utilizzarlo come ingresso del progetto software.

Per cominciare selezionare dal menù a tendina il percorso *Edit* → *Insert Symbol*. Nella schermata che compare si possono scegliere i vari blocchi in libreria. Quindi selezionare *libraries* → *primitives* → *pin* → *input*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.6.

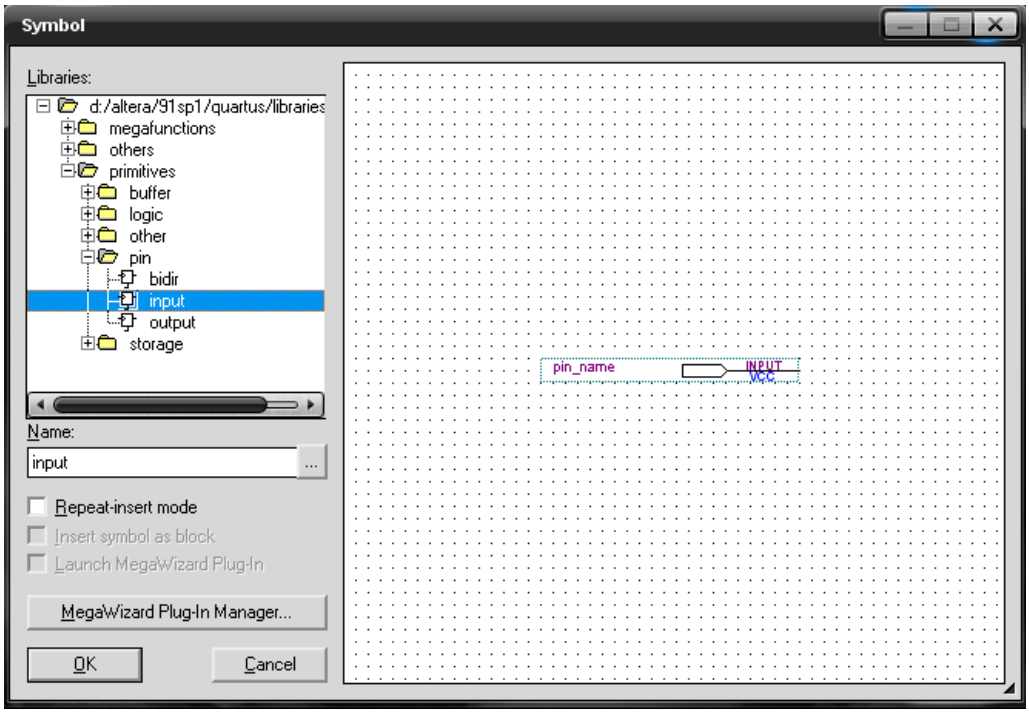

Fig. 3.6: Pin di ingresso

A questo punto selezionare *OK* ed inserire il pin di ingresso sullo schematico. Facendo doppio click sul simbolo lo si può rinominare a piacere. Infine manca l'assegnazione al pin fisico dell'FPGA, per la quale si rimanda alla sottosezione 3.5.12.

Con le operazioni appena effettuate, si ha a disposizione nello schematico il pin corrispondente al Clock interno dell'FPGA a 50Mhz. Ma per il nostro progetto servono dei clock inferiori, e quindi si deve inserire un blocco di condizionamento del Clock a 50Mhz. In particolare si inserirà un PLL (Phase Locked Loop) detto anello ad aggancio di fase. Tale blocco è in grado di fornire frequenze differenti da quella del core dell'FPGA.

Per usufruire di questa funzionalità selezionare nuovamente *Edit* → *Insert Symbol*, ma stavolta scegliere il percorso *libraries* → *megafunction* → *IO* → *altpll*. Nella schermata riportata in Fig. 3.7 scegliere *OK*. Mentre in quella successiva scegliere il nome da dare al blocco (per esempio *Clock*) e premere *Next*. In questo modo comparirà la finestra di configurazione vera e propria riportata in Fig. 3.8. Nella quale basta impostare la frequenza del clock di ingresso (Clock interno all'FPGA) che nel caso in esame è pari a 50Mhz, e premere *Next*. Nella schermata seguente deselezionare tutto e proseguire. Proseguire senza modificare nulla anche nelle seguenti tre videate. Infine si giunge alla schermata di selezione dei clock desiderati in uscita.

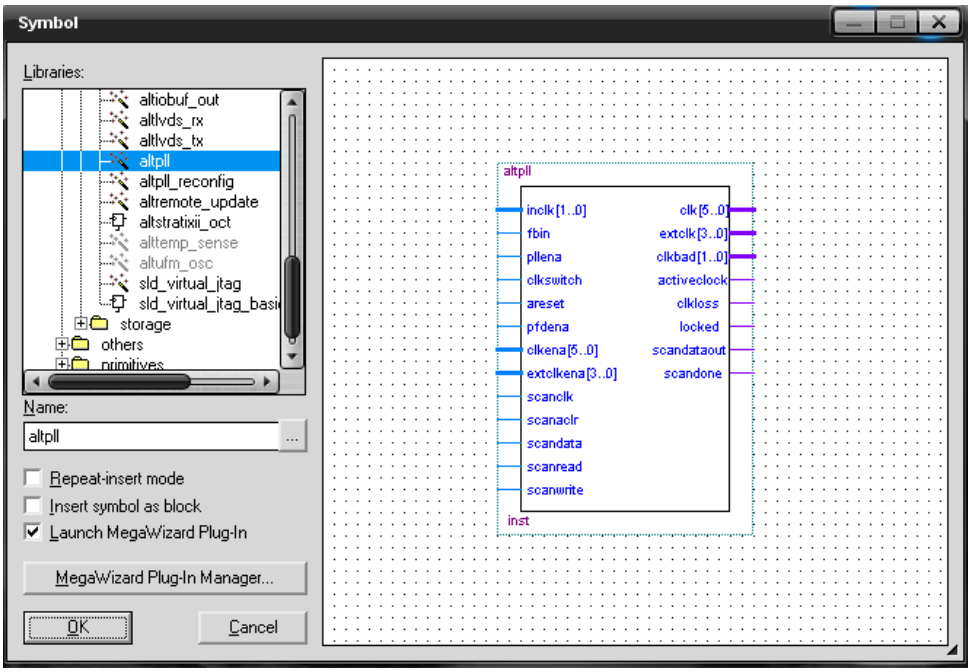

Fig. 3.7: Selezione Blocco PLL

Nel progetto è stato impostato il *clk_c0* a 20Mhz. Semplicemente impostando la frequenza in questione alla voce *Enter output clock frequency:* e premendo il pulsante *Next*.

Mentre il *clk_c0* è attivo per default, gli altri clock devono essere attivati. Per farlo basta spuntare la voce *Use this Clock*. Il *clk_c1* è stato impostato anch'esso a 20Mhz, in più è stato introdotto uno sfasamento pari a 6ns, impostando nella voce: *Clock phase shift* il valore 6, dopo aver selezionato nel menù a tendina a fianco, l'unità di misura *ns*.

Per ultimo è stato attivato anche il *clk_c2*, ed è stato impostato a 0,02Mhz. Per ultimare la procedura basta premere due volte su *Finish* e posizionare il blocco PLL sullo schematico. Infine non resta che collegare il pin di input corrispondente al clock dell'FPGA all'ingresso del blocco PLL appena creato.

La generazione del nuovo blocco PLL porta alla creazione di alcuni file all'interno della cartella del progetto:

- *Clock.v* = File di implementazione Verilog del funzionamento del blocco
- *Clock_bb.v* = File di implementazione Verilog di ingressi e uscite del blocco
- *Clock.ppf* = File di configurazione dei pin
- *Clock.bsf* = File contenente lo schematico del blocco

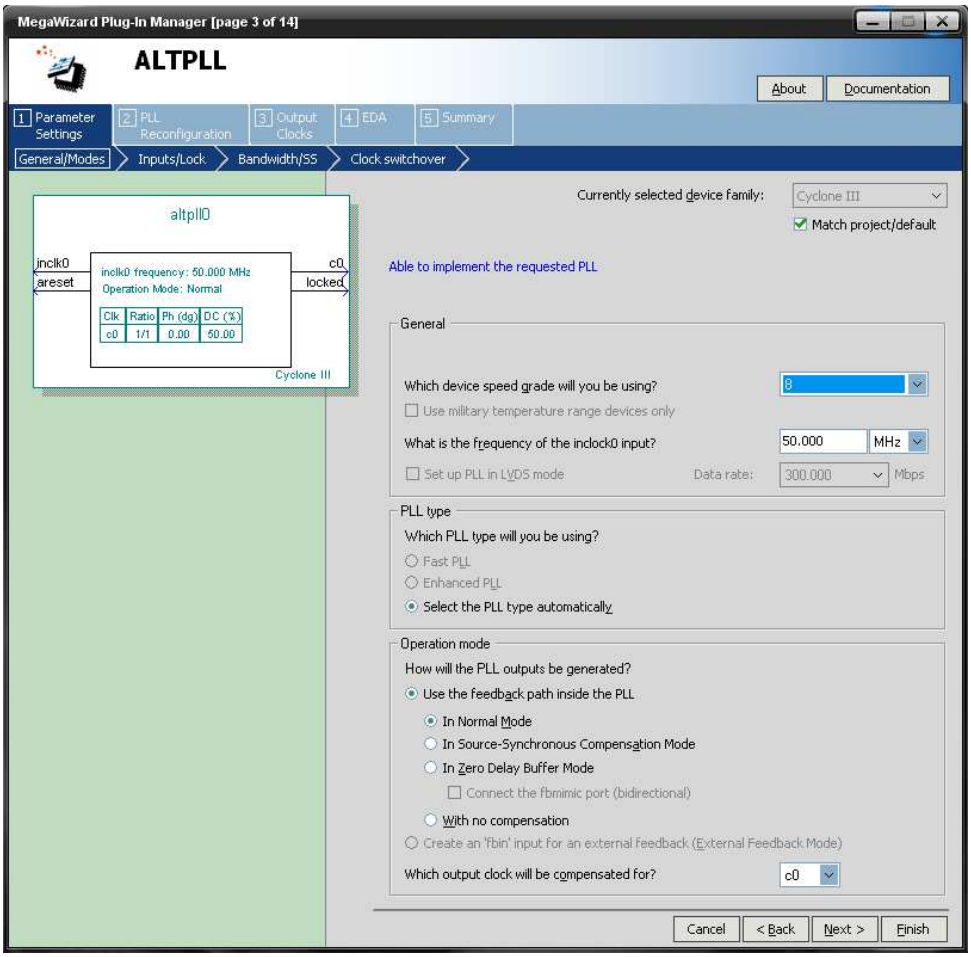

Fig. 3.8: Configurazione del PLL

- *Clock.qip* = File di archiviazione (nel quale sono salvati i nomi degli altri quattro file)

Nel progetto in esame si utilizzano i tre tipi di clock creati come segue.

Il *clk_c0* a 20Mhz viene utilizzato per impostare i clock dei convertitori AD e DA della scheda di acquisizione analogica/digitale (come spiegato più approfonditamente nella sottosezione 3.5.2). Il *clk_c1* a 20Mhz con sfasamento di 6ns viene utilizzato per impostare i clock degli accumulatori e viene inoltre utilizzato nello schema di creazione del segnale *Sload* per gli accumulatori (come spiegato più approfonditamente nella sottosezione 3.5.4). Per capire il motivo dell'introduzione dei 6ns di sfasamento si rimanda alla fine della sottosezione 2.4.2. Il *clk_c2* a 20Khz viene utilizzato, come clock di campionamento dello strumento di simulazione `Signal Tap` fornito all'interno del software `Quartus II` (come spiegato più approfonditamente nella sezione 4.2).

Il blocco PLL ottenuto viene riportato in Fig. 3.9.

3.5.2 Configurazione del clock dei convertitori AD e DA

Nella prima parte di questa sottosezione verrà mostrata la procedura di inserimento nello schematico di un pin di output. Analogamente alla spiegazione del pin di ingresso, proposta nella sottosezione 3.5.1; un pin di uscita nello schematico non è altro che un puntatore ad un pin fisico presente nell'FPGA. Creando, con il procedimento spiegato a breve, un pin di uscita, e collegandolo a monte ad un opportuno segnale, si chiede al programma di far

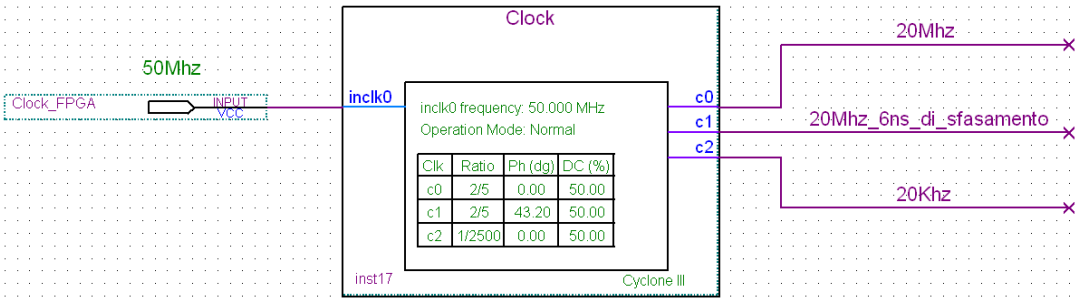

Fig. 3.9: Blocco PLL per la generazione dei segnali di clock

uscire il segnale appena menzionato attraverso il pin fisico dell'FPGA, puntato dal pin di output dello schematico software.

Per cominciare selezionare dal menù a tendina il percorso *Edit* → *Insert Symbol*. Quindi selezionare *libraries* → *primitives* → *pin* → *output*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.10.

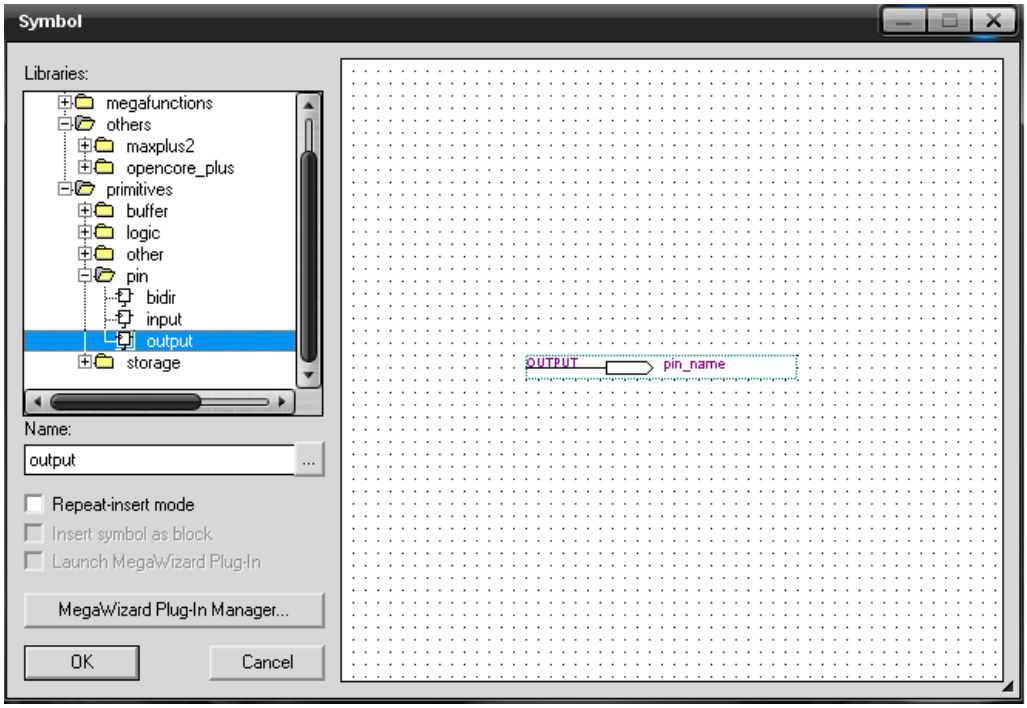

Fig. 3.10: Pin di uscita

A questo punto selezionare *OK* ed inserire il pin di uscita sullo schematico. Facendo doppio click sul simbolo lo si può rinominare a piacere.

Nello schematico sono stati creati sei pin di uscita, che impostano i corrispondenti pin di ingresso del clock, presenti nella scheda di acquisizione analogica/digitale. Brevemente vengono approfondite le varie corrispondenze.

Nelle sottosezioni 2.4.2 e 2.4.3 si è detto che i clock dei convertitori AD e DA vengono fornito via software dall'FPGA. Di seguito riportiamo nella Tab. 3.2 le corrispondenze tra i pin di clock all'interno del sistema totale. In particolare la prima colonna è relativa ai pin posizionati nel connettore HSMC sul lato relativo alla scheda FPGA, che sono adibiti al trasporto del segnale di clock programmato via software. Nella seconda colonna sono elencati i pin corrispondenti posizionati nel connettore HSMC sul lato della scheda di acquisizione analogica/digitale. Infine nella terza colonna sono elencati i pin corrispondenti

sui componenti ADC e DAC. Le informazioni riportate in Tab. 3.2 sono state ricavate da [6], [7], [8] e [13].

HSMC scheda FPGA	HSMC scheda di conversione A/D	ADC e DAC
HSMC_CLKOUT_n2 (pin 157)	PLL_OUT_ADC0 (pin 3)	CLK_A (pin 63 ADC)
HSMC_CLKOUT_p2 (pin 155)	PLL_OUT_ADC1 (pin 5)	CLK_B (pin 18 ADC)
HSMC_CLKOUT_n1 (pin 97)	PLL_OUT_DAC0 (pin 63)	CLK1 (pin 18 DAC)
HSMC_CLKOUT_p1 (pin 95)	PLL_OUT_DAC1 (pin 65)	CLK2 (pin 19 DAC)
HSMC_TX_p7 (pin 89)	DA_WRTA (pin 71)	WRT1 (pin 17 DAC)
HSMC_RX_p7 (pin 90)	DA_WRTB (pin 72)	WRT2 (pin 20 DAC)

Tab. 3.2: Corrispondenze Hardware dei pin di clock

Nello schematico sono stati creati sei pin di output utilizzati per settare i clock dei convertitori AD e DA della scheda Terasic_THDB_ADA. Sono stati nominati come la prima colonna di Tab. 3.2. Quindi corrispondentemente ai nomi relativi alla scheda FPGA. Ciò non è una regola e non è vincolante. Infatti è la successiva assegnazione dei pin software ai pin fisicamente presenti nell'FPGA che realizza l'effettiva corrispondenza tra schematico e hardware. A tale procedura è dedicata la sottosezione 3.5.12.

In ingresso ai sei pin di uscita è stato collegato il colck a 20Mhz, la cui realizzazione è stata approfondita in sezione 3.5.1. In Fig. 3.11 vengono riportati i sei pin di uscita creati nello schematico.

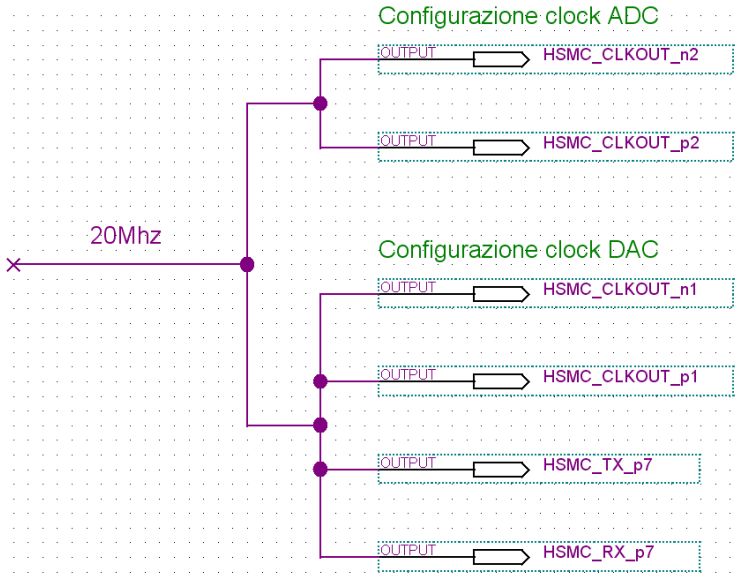

Fig. 3.11: Pin di clock per i convertitori AD e DA

3.5.3 Pin di configurazione dei convertitori AD e DA

Nella prima parte di questa sottosezione verrà mostrata la procedura di inserimento nello schematico di una costante. Per cominciare selezionare dal menù a tendina il percorso *Edit* → *Insert Symbol*. Quindi selezionare *libraries* → *megafunction* → *gates* → *lpm_constant*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.12.

A questo punto selezionare *OK*. Nella schermata successiva scegliere il nome da dare al blocco e premere *Next*. Nella videata che compare si devono scegliere in ordine: la

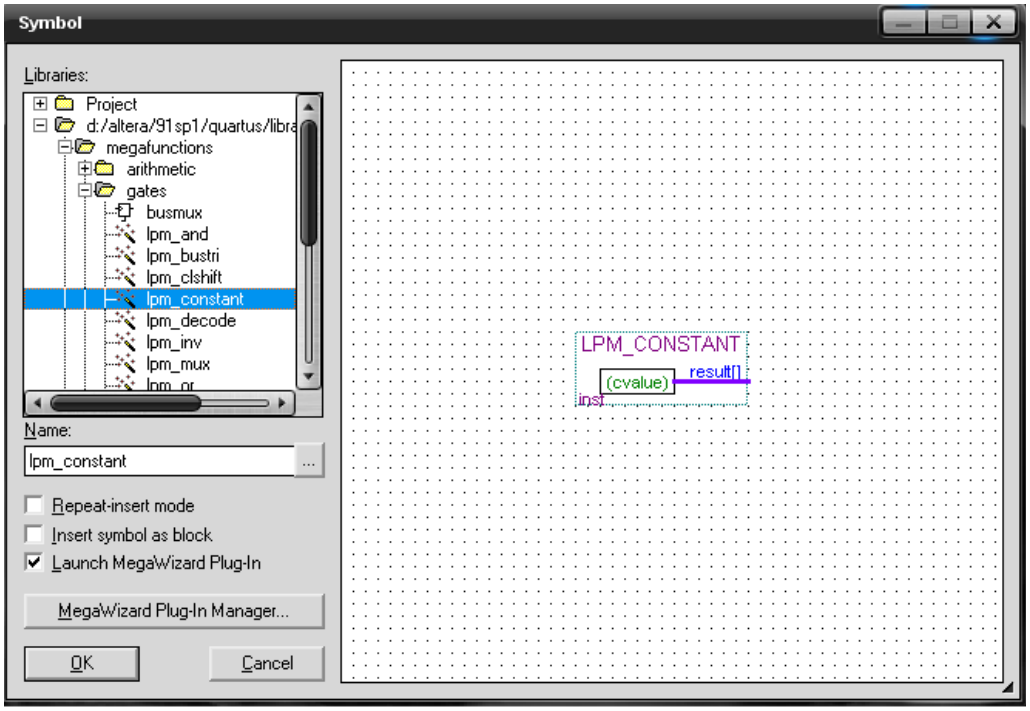

Fig. 3.12: Blocco costante

profondità del dato di uscita, (in questo caso è stato scelto 1bit) ed il valore del dato di uscita, posticipato dal formato numerico selezionabile dal menù a tendina. Nel progetto è stato selezionato il formato binario. E sono state create due costanti una a 0 e una a 1. Successivamente premere *Finish* e posizionare il blocco costante nello schematico. La generazione del nuovo blocco costante, porta alla creazione (nella cartella del progetto) di file analoghi a quelli riportati in sottosezione 3.5.1, riguardanti la creazione di un blocco PLL. Da qui in poi, tale informazione sarà omessa, e si darà per scontato che ogni volta che si creerà un nuovo blocco nello schematico, nella cartella del progetto, avverrà la generazione dei file che contengono le informazioni del blocco stesso.

La realizzazione delle due costanti 0 e 1 sarà utile a breve.

Nella sottosezione 2.4.2 si è detto che i pin OEP_A e OEP_B (Output Enable Pin) servono per abilitare i canali di uscita dei due convertitori AD presenti nella scheda di acquisizione analogica/digitale. Per far questo devono essere settati entrambi al valore logico basso. Prima di vedere come è stata realizzata tale configurazione nello schematico; riportiamo in Tab. 3.3 (analogamente a quanto fatto nella sottosezione 3.5.2) le corrispondenze hardware dei pin in questione nel sistema totale.

HSMC scheda FPGA	HSMC scheda di conversione A/D	ADC
HSMC_RX_p9 (pin 108)	ADC_OEA (pin 54)	OEP_A (pin 59)
HSMC_RX_n9 (pin 110)	ADC_OEB (pin 52)	OEP_B (pin 22)

Tab. 3.3: Corrispondenze Hardware dei pin di abilitazione delle uscite degli ADC

Nella sottosezione 2.4.3 si è detto che il pin MODE deve essere settato al valore logico alto per abilitare la modalità di funzionamento 'dual-port' del convertitore DA. Prima di vedere come è stata realizzata tale configurazione nello schematico; riportiamo in Tab. 3.4 le corrispondenze hardware del pin in questione nel sistema totale.

HSMC scheda FPGA	HSMC scheda di conversione A/D	DAC
HSMC_TX_n7 (pin 91)	DA_MODE (pin 69)	MODE (pin 48)

Tab. 3.4: Corrispondenze Hardware del pin MODE

Le informazioni riportate in Tab. 3.3 e in Tab. 3.4 sono state ricavate da [6], [7], [8] e [13].

Nello schematico sono stati creati tre pin di output (come spiegato in sottosezione 3.5.1) utilizzati per abilitare i canali di uscita dell'ADC e per attivare la modalità 'dual-port' del DAC, come spiegato in questa sottosezione. Sono stati nominati in maniera corrispondente alla prima colonna di Tab. 3.3 e di Tab. 3.4. Quindi corrispondentemente ai nomi relativi alla scheda FPGA. Ciò (come già detto in sottosezione 3.5.2) non è una regola e non è vincolante. Infatti è la successiva assegnazione dei pin software ai pin fisicamente presenti nell'FPGA che realizza l'effettiva corrispondenza tra schematico e hardware. A tale procedura è dedicata la sottosezione 3.5.12.

In ingresso ai due pin di abilitazione dei canali di uscita dell'ADC è stato collegato il blocco costante con valore logico basso. In ingresso al pin di attivazione della modalità 'dual-port' del DAC è stato collegato il blocco costante con valore logico alto. In Fig. 3.13 vengono riportati i tre pin di uscita creati nello schematico.

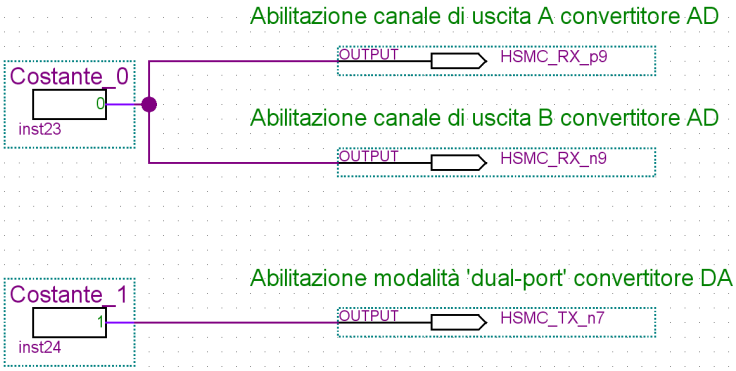

Fig. 3.13: Pin di configurazione dei convertitori AD e DA

3.5.4 Generazione dei segnali di sincronismo

Nella realizzazione software dell'algoritmo di calcolo della media mobile, (illustrato nella sottosezione 3.4.2) sono richiesti numerosi segnali di sincronismo, che verranno forniti ai vari blocchi dello schematico, col fine di organizzare temporalmente le varie operazioni.

Prima di tutto, occorre prelevare il segnale di sincronismo della PWM, condizionato in maniera opportuna (vedi sottosezione 2.3.2) e iniettato nel pin dell'FPGA `HSMC_RX_p8` (come già detto in sezione 2.5).

Per far questo basta introdurre nello schematico un nuovo pin di input (vedi sottosezione 3.5.1), nominarlo a piacere, e, come sarà approfondito nella sottosezione 3.5.12, collegarlo al pin reale presente nell'FPGA.

Il segnale di sincronismo PWM prelevato al pin `HSMC_RX_p8` dell'FPGA è un'onda quadra a 10Khz (quindi con periodo pari a $100\mu s$).

Il blocco accumulatore (vedi sottosezione 3.5.7) è attivo se in ingresso alla porta *sload* è presente un segnale con valore logico basso. Nelle esigenze di progettazione si desidera che gli accumulatori nei due rami dello schematico, rimangano attivi nei periodi PWM in maniera alternata. Per realizzare ciò si deve condizionare il segnale di sincronismo PWM, in modo da dimmezzarne la frequenza, e successivamente sdoppiare il sincronismo con

frequenza dimezzata, realizzando un Sincronismo diretto (*Sync_Diretto*) e un Sincronismo negato (*Sync_Negato*). Ognuno dei quali andrà ad agire nel canale di competenza. La realizzazione software consiste nei seguenti passaggi.

Il segnale con perido doppio rispetto al segnale di sincronismo PWM (e quindi con periodo pari a $200\mu s$) è stato ottenuto introducendo nello schematico un blocco che realizza un Flip Flop edge triggered sensibile solamente ai fronti di salita.

Per realizzare questo blocco selezionare il percorso *Edit* → *Insert Symbol*. Quindi selezionare *libraries* → *megafunction* → *storage* → *lpm_tff*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.14.

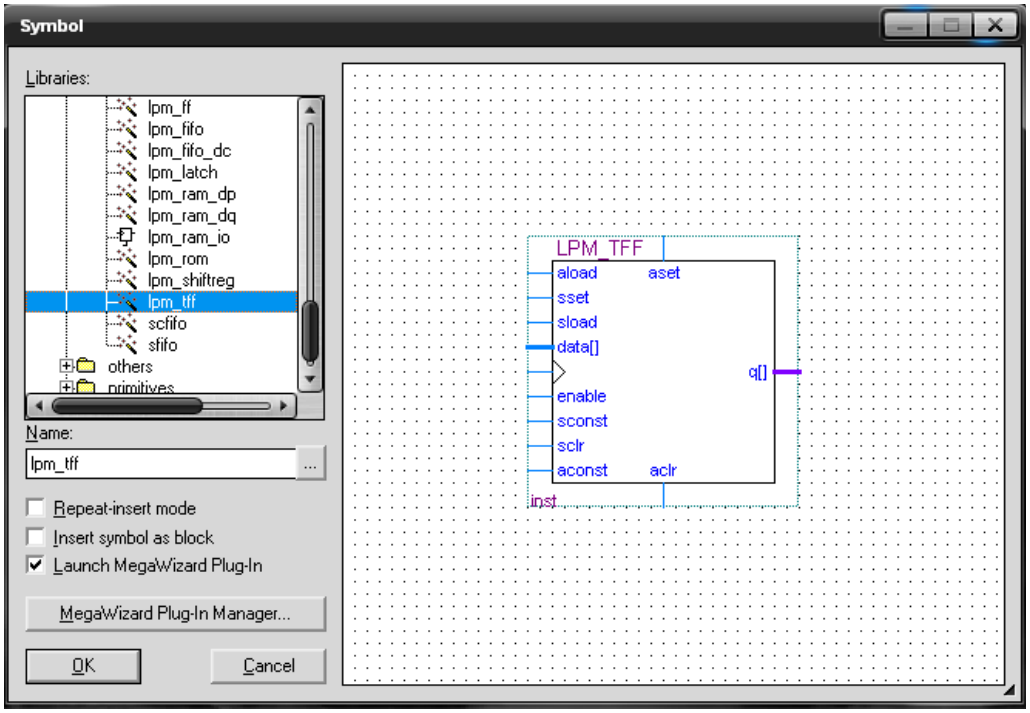

Fig. 3.14: Blocco Flip Flop edge triggered

Premere *OK* e sulla schermata che appare scegliere il nome da dare al blocco e premere *Next*. Nel menù a tendina di fianco alla domanda: *How many flipflops do you want?* selezionare 1. In questo modo si imposta l'uscita del Filp Flop ad 1bit. Lasciare selezionato il *T flipflop* e premere *Finish*.

Come, anticipato sopra, è necessario ottenere due segnali (*Sync_Diretto* e *Sync_Negato*) complementari, in modo che l'uno attivi l'accumulatore di un canale mentre contemporaneamente l'altro interdisce l'accumulatore sull'altro canale e viceversa. Per ottenere ciò è sufficiente sdoppiare il canale col sincronismo e negare uno dei due rami ottenuti, introducendo una semplice operazione di negazione.

Per introdurre il Blocco *NOT* seguire i seguenti passaggi. Selezionare il percorso *Edit* → *Insert Symbol*. Quindi selezionare *libraries* → *primitives* → *logic* → *not*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.15.

Le realizzazione sullo schematico dei segnali (*Sync_Diretto* e *Sync_Negato*) è riportata in Fig. 3.16, mentre il risultato dell'elaborazione del segnale di Sincronismo PWM è riportato in Fig. 3.17.

I segnali così creati verranno iniettati (in maniera duale sui due canali):

- in ingresso ai blocchi di memorizzazione dei dati di uscita (vedi sottosezione 3.5.9)
- in ingresso al blocco di selezione del canale di uscita (solo il *Sync_Diretto*) (vedi sottosezione 3.5.10)

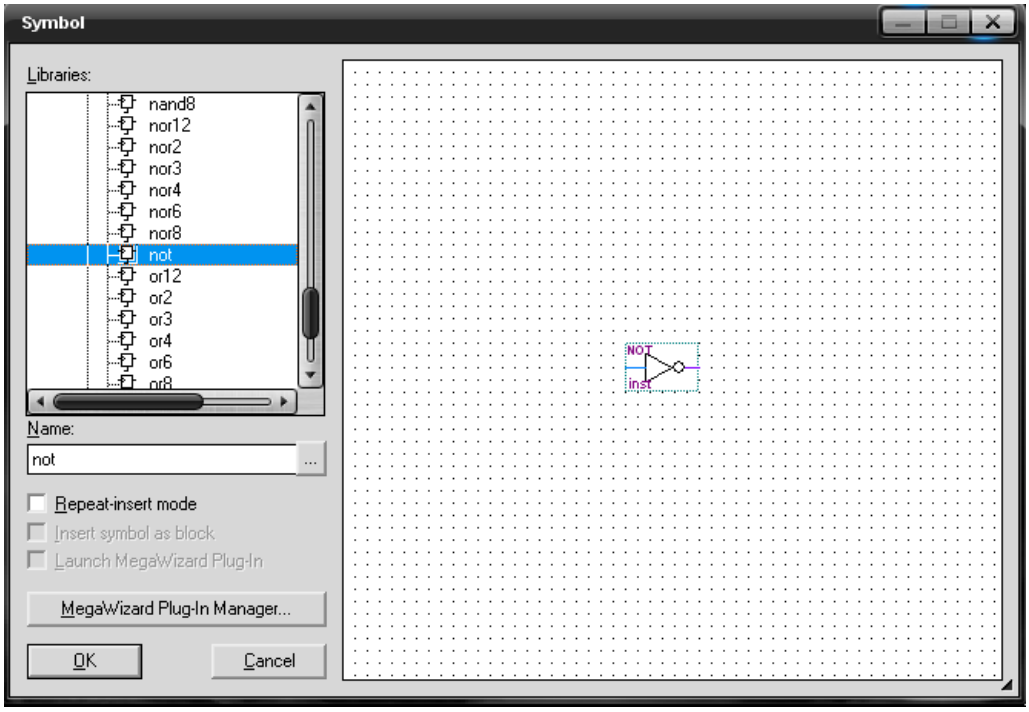

Fig. 3.15: Blocco di negazione

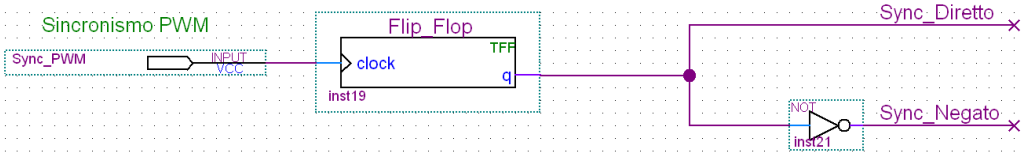

Fig. 3.16: Schematico di creazione dei segnali *Sync_Diretto* e *Sync_Negato*

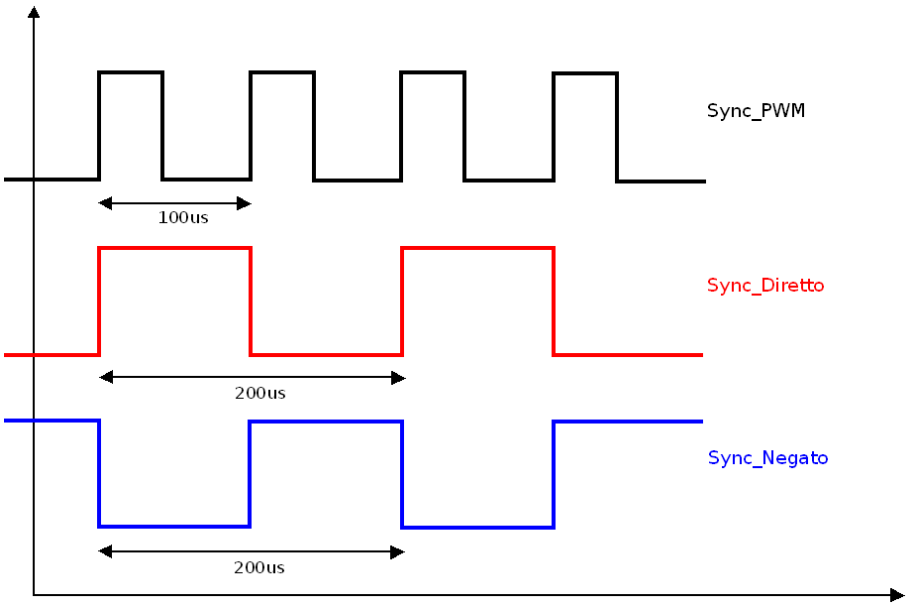

Fig. 3.17: Elaborazione sincronismo di ingresso

- in ingresso al circuito di creazione dei segnali di comando *Sload_Diretto* e *Sload_Negato* per gli accumulatori, che verrà spiegato di seguito.

Come già detto in sottosezione 3.4.2, alla fine di un periodo di accumulazione ci si aspetta che l'accumulatore che ha appena finito di sommare i dati di ingresso si interdisca, non prima che il dato di uscita venga salvato all'interno del blocco di memorizzazione (vedi sottosezione 3.5.9) presente in uscita dall'accumulatore stesso.

Si fa notare che l'accumulatore si interdisce con un segnale con valore logico alto in ingresso alla porta *sload*, mentre il blocco di memorizzazione della sommatoria di uscita salva il valore finale dell'accumulazione se in ingresso alla porta *gate* il segnale ha valore logico basso.

Focalizzando l'attenzione su uno dei due rami (i ragionamenti sono duali per altro), l'operazione di sincronizzazione tra accumulatore e blocco di memorizzazione è stata inizialmente realizzata iniettando direttamente il segnale *Sync_Negato* nella porta *sload* dell'accumulatore e il segnale *Sync_Diretto* nella porta *gate* del blocco di memorizzazione. Teoricamente la contemporanea attivazione delle porte *sload* e *gate* dovrebbe portare al risultato finale sperato. Dall'esperienza in laboratorio, si è visto che ciò si verificava per la maggior parte dei periodi di accumulazione, ma purtroppo per un numero non trascurabile di operazioni si è osservato che accadeva quanto riportato di seguito.

A causa della propagazione diversa, all'interno dell'FPGA, dei segnali *Sync_Diretto* e *Sync_Negato*; a volte il segnale *Sync_Negato* giungeva all'accumulatore prima di quanto impiegasse il segnale *Sync_Diretto* a raggiungere il blocco di memorizzazione. Quindi l'accumulatore resettava il valore di uscita prima del salvataggio dello stesso nel blocco di memorizzazione. Producendo dei dati di uscita non corretti.

Per ovviare a tale inconveniente è stato sufficiente far iniziare la fase di reset dell'accumulatore con un certo tempo di ritardo rispetto alla fase di salvataggio del dato da parte del blocco accumulatore. Tale fine è stato ottenuto creando due nuovi segnali *Sload_Diretto* e *Sload_Negato*(uno per ciascun ramo) a partire dai segnali *Sync_Diretto* e *Sync_Negato*.

In particolare il condizionamento introdotto dallo schema a blocchi, che fra poco verrà approfondito, consiste nel ritardare il solo fronte di salita dei segnali *Sync_Diretto* e *Sync_Negato*; essendo appunto tale fronte quello che ordina il reset agli accumulatori. Mentre il fronte di discesa non deve essere alterato, essendo quello che ordina l'inizio dell'accumulazione.

Dalle prove in laboratorio si è visto che l'introduzione di un ritardo al reset degli accumulatori pari a circa $13\mu s$, è sufficiente a rendere il progetto perfettamente funzionante (eliminando gli errori causati dalla non perfetta sincronizzazione delle operazioni).

La realizzazione pratica è descritta di seguito, dove, per praticità, ci riferiremo al condizionamento del segnale *Sync_Diretto* (per il segnale *Sync_Negato* i ragionamenti sono duali).

Il blocco principale del circuito di condizionamento del segnale *Sync_Diretto* consiste in un contatore. Il conteggio viene abiliatato ogni qual volta si verifica un fronte di salita del segnale *Sync_Diretto*. Per realizzare ciò è stato iniettando tale segnale in ingresso alla porta *clk_en* (Clock Enable) del contatore . Vicevera, se il segnale *Sync_Diretto* ha valore logico basso, il contatore è interdetto e la sua uscita rimane costante a zero. Ciò è realizzato collegando il segnale *Sync_Diretto* NEGATO (utilizzando un blocco di negazione già visto in questa sottosezione) nella porta di ingresso *aclr* del contatore. Quindi il valore logico alto attiva la porta *aclr* che pone il contatore in interdizione e imposta la sua uscita a zero.

Al contatore viene fornito il clock a 20Mhz con i 6 ns di sfasamento (vedi sottosezione 2.4.2) ricavato in sottosezione 3.5.1, collegando tale segnale alla porta *clock* del contatore stesso.

Il registro interno che mantiene memoria della progressiva accumulazione ha una profondità pari a 9bit. Quando il MSB di tale registro raggiunge il valore logico alto, un blocco realizzato in Verilog HDL, chiamato *settaggio*, pone la sua uscita (normalmente a zero) a uno. L'uscita del blocco *settaggio* è collegata alla porta di ingresso *aset* del contatore. Tale porta forza tutti i bit di uscita del contatore al valore logico alto. Un altro blocco realizzato in Verilog HDL, chiamato *selettore_MSB* riceve in ingresso l'uscita del contatore a 9bit e pone in uscita solamente il MSB.

Ricapitolando, con questo sistema quando il *Sync_Diretto* ha valore logico basso, l'uscita del circuito di condizionamento ha valore logico anch'esso basso. Quindi il sistema risulta trasparente al valore logico basso (il contatore è disabilitato e l'uscita permane a zero). Invece quando il *Sync_Diretto* passa al valore logico alto, il contatore si attiva e il circuito totale crea un fronte di salita ritardato rispetto al fronte di ingresso, del tempo impiegato dal contatore a far sì che il MSB raggiunga il valore logico alto.

Il contatore per raggiungere il numero N in base dieci, deve effetture N conteggi. Il numero decimale corrispondente a far sì che il nono bit raggiunga il valore logico alto (100000000 in binario) equivale a $2^8 = 256$. Pertanto il contatore impiega 256 conteggi per impostare al valore logico alto il MSB. Quindi, ricordando che la frequenza di lavoro del contatore è pari a 20Mhz, il ritardo complessivo è dato dalla 3.8.

$$Ritardo = 256 \cdot \frac{1}{20Mhz} = 12,8\mu s \tag{3.8}$$

Di seguito viene riportata l'implementazione software dei vari blocchi.

Per la creazione del blocco contatore selezionare il percorso *Edit* → *Insert Symbol*. Quindi selezionare *libraries* → *megafunctions* → *arithmetic* → *lpm_counter*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.18.

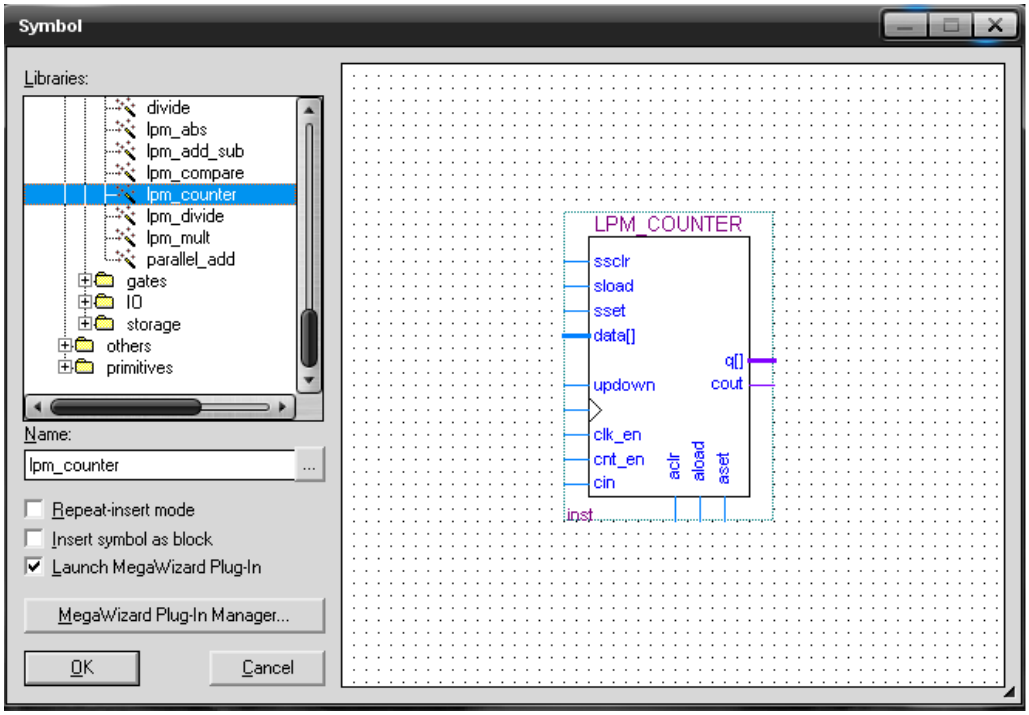

Fig. 3.18: Blocco contatore

Selezionare *OK*, nella schermata successiva scegliere il nome del blocco contatore, e premere *Next*. Nel menù a tendina di fianco alla domanda: *How wide should the 'q' output bus be?* selezionare 9. In questo modo si imposta l'uscita del contatore a 9bit. Lasciare spuntata la voce *Up only*, in modo che il contatore conti solo verso l'alto, e premere *Next*.

Nella successiva videata lascire le selezioni di default, spuntare la voce *Clock Enable* e proseguire. Nella sezione *Asynchronous Input* spuntare le voci *Clear* e *Set* e lasciare la selezione della voce *Set to all 1's*. Infine si può premere *Finish* e posizionare il blocco contatore creato sullo schematico.

Per la creazione del blocco *settaggio* occorre prima di tutto creare un nuovo file Verilog HDL. Per far questo selezionare *File* → *New*. Nella schermata che compare scegliere *Verilog HDL file* e premere *OK*. In questo modo verrà aperta una nuova finestra di programmazione nella quale si deve scrivere il codice che caratterizzerà il blocco *settaggio*. Di seguito viene riportato il codice Verilog HDL del blocco *settaggio*.

```
module settaggio (in,set);      %nome del modulo e definizione di ingressi e uscite
output reg set;                 %definizione del registro di uscita ad un bit
input [8:0] in;                 %definizione dell'ingresso a 9bit
always                          %definizione di un ciclo infinito
begin                           %inizio del ciclo
%se il MSB dell'ingresso 'in' = 1 → 'set'=1
if (in[8])
set = 1;
else
%se il MSB dell'ingresso 'in' = 0 → 'set'=0
set = 0;
end                             %fine del ciclo
endmodule                       %fine del modulo
```

Una volta completato il codice, salvare il file nella directory del progetto, selezionando *File* → *Save As*; nel progetto è stato scelto il nome *settaggio.v*. Infine selezionare il percorso *File* → *Create/Update* → *Create Symbol Files for Current File*. In questo modo viene creato il blocco relativo al codice salvato. Ora si può chiudere la finestra col codice Verilog HDL, e non resta altro che introdurre nello schematico il blocco appena creato. Per far questo selezionare *Edit* → *Insert Symbol* e stavolta, non posizionarsi sulla libreria del `Quartus II`, ma selezionare la cartella *Project*, all'interno della quale sono presenti tutti i blocchi creati per il progetto. Scegliere il nome del blocco appena creato, premere *OK* e posizionare il blocco sullo schematico.

La creazione del blocco *selettore_MSB* è del tutto analoga a quella del blocco *settaggio*, per cui non ci si soffermerà oltre; si riporta semplicemente di seguito il codice Verilog HDL del blocco *selettore_MSB*.

```
module selettore_MSB (in,out);   %nome del modulo e definizione di ingressi e uscite
output reg out;                  %definizione del registro di uscita ad un bit
input [8:0] in;                  %definizione dell'ingresso a 9bit
always                           %definizione di un ciclo infinito
begin                            %inizio del ciclo
out = in[8];                     %l'uscita è uguale al MSB dell'ingresso
end                              %fine del ciclo
endmodule                        %fine del modulo
```

In Fig. 3.19 è riportatata la realizzazione del circuito di creazione dei segnali *Sload_Diretto* e *Sload_Negato*. In Fig. 3.20 sono riportati i segnali *Sload_Diretto* e *Sload_Negato* risultanti dall'elaborazione dei segnali *Sync_Diretto* e *Sync_Negato*.

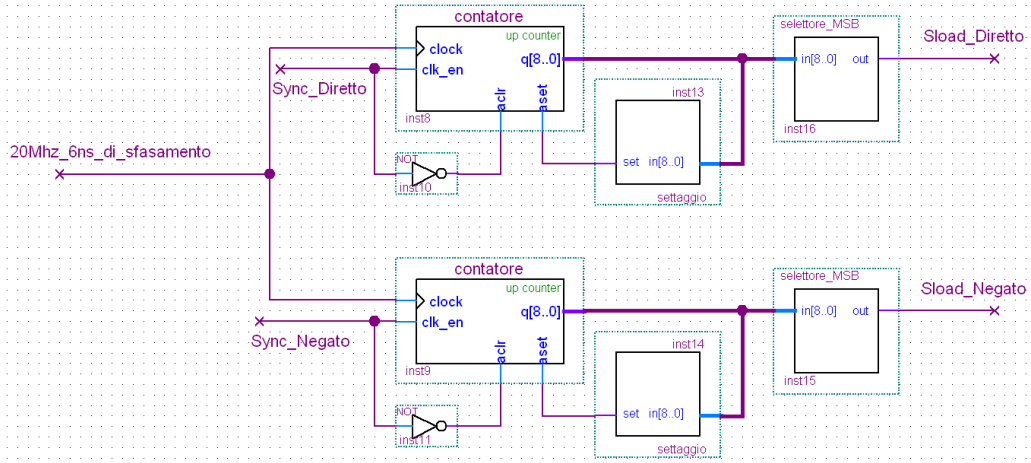

Fig. 3.19: Schematico di creazione dei segnali *Sload_Diretto* e *Sload_Negato*

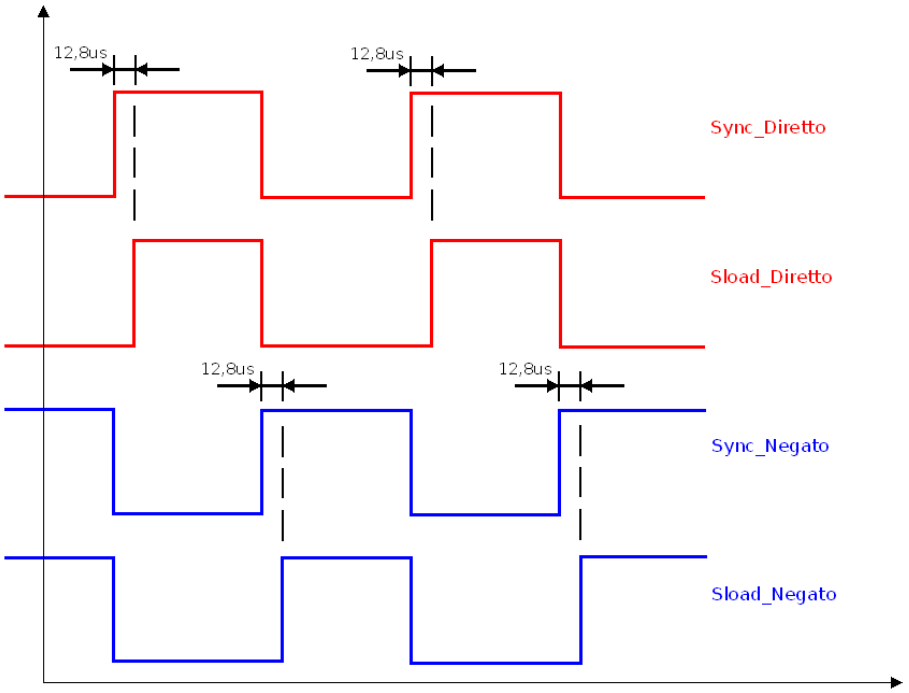

Fig. 3.20: Elaborazione dei segnali *Sync_Diretto* e *Sync_Negato*

3.5.5 Blocco di ingresso del segnale digitale a 14bit

Nella sottosezione 3.5.1 è stato spiegato come introdurre nello schematico un blocco di ingresso, quindi con il medesimo procedimento, è necessario introdurre nello schematico un blocco di input che, come sarà illustrato in sottosezione 3.5.12, sarà poi collegato ai pin dell'FPGA adibiti al trasporto dei dati digitalizzati dal convertitore AD. Una volta posizionato il blocco di ingresso nello schematico, bisogna nominarlo. Per indicare che esso farà riferimento a 14 pin di ingresso, il nome del blocco deve essere del tipo *Input_A[13..0]*; dove *Input_A* è un nome a piacere, mentre la designazione *[13..0]* sta appunto ad indicare, che il pin fa riferimento a 14 pin numerati da 0 a 13. In Fig. 3.21 si riporta il blocco di ingresso del canale A creato sullo schematico.

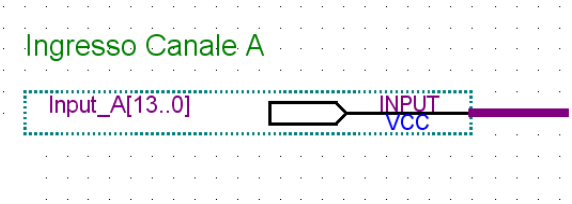

Fig. 3.21: Blocco di ingresso del canale A

3.5.6 Conversione da Offset Binary a Complemento a Due

Come visto in sottosezione 3.4.2, i dati di ingresso vengono subito trasformati da Offset Binary in Complemento a Due.

La creazione di un blocco Verilog HDL è stata già affrontata nella sottosezione 3.5.4. Quindi ci si limita a riportare il codice Verilog HDL del blocco di conversione da Offset Binary a Complemento a Due.

```
module conversione_C2 (in, out);    %nome del modulo e definizione di ingressi e uscite
output reg [13:0] out;              %definizione del registro di uscita a 14 bit
input [13:0] in;                    %definizione dell'ingresso a 14bit
always                              %definizione di un ciclo infinito
begin                               %inizio del ciclo
%tutti i bit di uscita sono uguali ai bit di ingresso tranne il MSB che viene negato
out[13] = !in[13];
out[12] = in[12];
out[11] = in[11];
out[10] = in[10];
out[9] = in[9];
out[8] = in[8];
out[7] = in[7];
out[6] = in[6];
out[5] = in[5];
out[4] = in[4];
out[3] = in[3];
out[2] = in[2];
out[1] = in[1];
out[0] = in[0];
end                                 %fine del ciclo
endmodule                           %fine del modulo
```

In Fig. 3.22 si riporta il blocco di conversione da Offset Binary a Complemento a Due realizzato sullo schematico.

3.5.7 Accumulatori

Come visto in sottosezione 3.4.2, il progetto ha bisogno di due accumulatori sullo stesso canale. Per creare un blocco accumulatore seguire i seguenti passaggi. Selezionare *Edit* → *Insert Symbol*. Quindi selezionare *libraries* → *megafunctions* → *arithmetic* → *altaccumulate*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.23.

Premere *OK*, nella schermata successiva scegliere il nome del blocco e premere *Next*. Nella videata che appare, alla voce *How wide should the 'data' input bus be?* selezionare

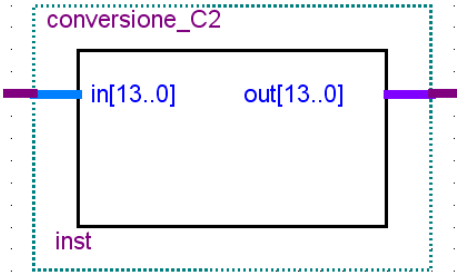

Fig. 3.22: Blocco di conversione da Offset Binary a Complemento a Due

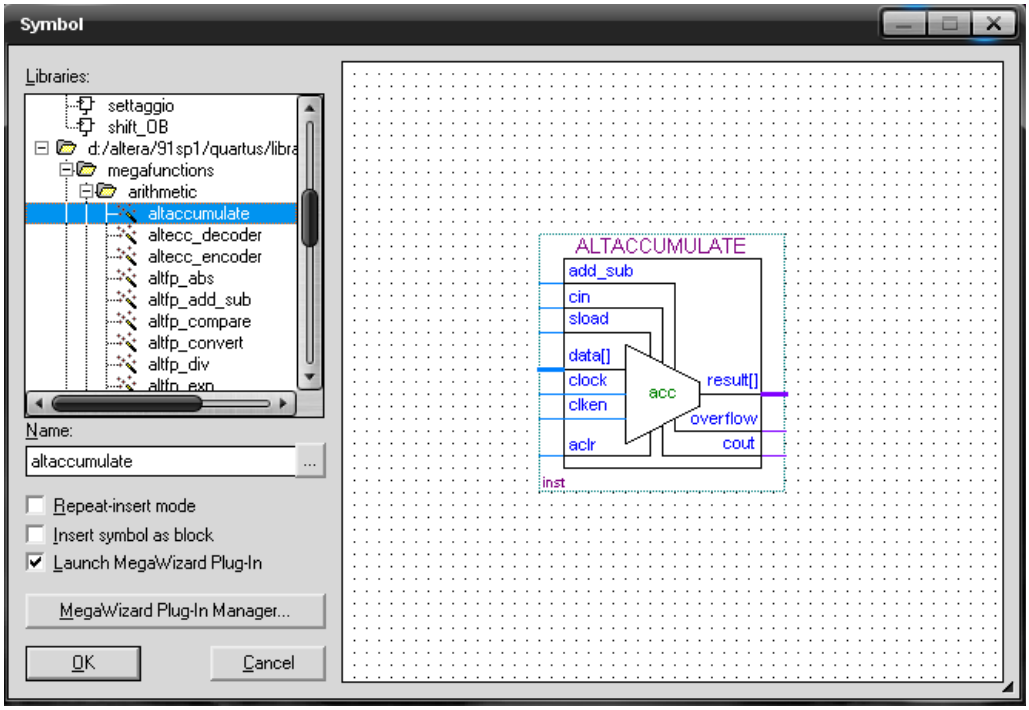

Fig. 3.23: Blocco accumulatore

dal menù a tendina *14bit*. In questo modo si imposta il bus di ingresso dell'accumulatore. Successivamente alla voce *How wide should the 'result' output bus be?* selezionare dal menù a tendina *25bit*. In questo modo si imposta il registro di uscita dell'accumulatore. Nella sezione *Data Rapresentation* spuntare la vose *Signed*. Così facendo l'accumulatore legge i dati in ingresso in Complemento a Due in maniera corretta, e, prima di effettuare la somma tra il nuovo dato in ingresso e il valore del registro di uscita, effettua l'estensione a 25bit del dato in ingresso a 14bit. Premendo *Next* si passa alla videata dove si deve spuntare la casella *Create a 'sload' port*; che, come si intuisce dal nome, serve per creare la porta *sload*. L'accumulaotre di un ramo, riceverà nella porta *sload* il segnale *Sload_Diretto*, mentre l'accumulaotre dell'altro ramo, riceverà nella porta *sload* il segnale *Sload_Negato*. Entrambi i segnali sono stati realizzati in sottosezione 3.5.4. Quando il segnale in ingresso alla porta *sload* rimane al valore logico basso l'accumulatore è attivo, viceversa, se tale segnale raggiunge il valore logico alto, porta l'accumulatore in interdizione e forza l'uscita al valore contenuto nella porta di ingresso.

In Fig. 3.24 si riporta il blocco accumulatore realizzato sullo schematico.

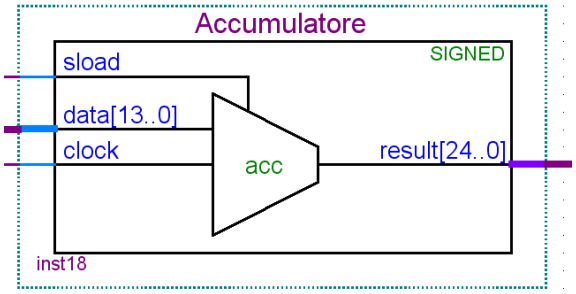

Fig. 3.24: Blocco accumulatore posizionato sullo schematico

3.5.8 Conversione da Complemento a Due ad Offset Binary e troncamento

I dati di uscita dall'accumulatore a 25bit in Complemento a Due devono essere riconvertiti a 14bit in Offset Binary per poter essere correttamente interpretati dal convertitore DA. Pertanto si deve realizzare un blocco che effettui tale conversione.

La creazione di un blocco Verilog HDL è stata già affrontata nella sottosezione 3.5.4. Quindi ci si limita a riportare il codice Verilog HDL del blocco di conversione da Complemento a Due a Offset Binary e troncamento da 25bit a 14bit.

```
module shift_OB (in, out);          %nome del modulo e definizione di ingressi e uscite
output reg [13:0] out;              %definizione del registro di uscita a 14bit
input [24:0] in;                    %definizione dell'ingresso a 25bit
always                              %definizione di un ciclo infinito
begin                               %inizio del ciclo
%shif logico a destra di 11 posizioni e negazione del MSB
out[13] = !in[24];
out[12] = in[23];
out[11] = in[22];
out[10] = in[21];
out[9] = in[20];
out[8] = in[19];
out[7] = in[18];
out[6] = in[17];
out[5] = in[16];
out[4] = in[15];
out[3] = in[14];
out[2] = in[13];
out[1] = in[12];
out[0] = in[11];
end                                 %fine del ciclo
endmodule                           %fine del modulo
```

In Fig. 3.25 si riporta il blocco di conversione da Offset Binary a Complemento a Due e troncamento da 25bit a 14bit, realizzato sullo schematico.

3.5.9 Memorizzazione dell'accumulazione

Il valore finale dell'accumulazione di ciascuna periodo PWM, deve essere salvato in un registro temporaneo, e successivamente portato in uscita. Tale funzione è ben gestita dal

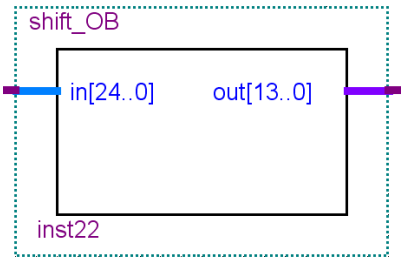

Fig. 3.25: Blocco di conversione da Complemento a Due a Offset Binary e troncamento

blocco *lpm_latch* presente all'interno della libreria del `Quartus II`. Tale blocco è dotato di una porta di comando chiamata *gate* che ha il seguente comportamento:

- se $gate = 0 \quad \Rightarrow \quad$ uscita = ingresso
- se $gate = 1 \quad \Rightarrow \quad$ uscita bloccata all'ultimo valore dell'ingresso

Pertanto sullo stesso ramo dello schematico l'accumulatore viene comandato con il segnale *Sload_Diretto* mentre l'*lpm_latch* viene comandato con il segnale *Sync_Negato*. In questo modo quando il segnale *Sload_Diretto* ha valore logico basso l'accumulatore è attivo, il segnale *Sync_Negato* ha valore logico alto, e quindi l'*lpm_latch* risulta trasparente ai dati in ingresso. Viceversa quando il segnale *Sload_Diretto* sale al valore logico alto l'accumulatore si interdisce e l'uscita risulta uguale al valore dell'ingresso. Pochi istanti prima che tutto ciò avvenga, il segnale *Sync_Negato* si porta al valore logico basso, e quindi l'uscita dell'*lpm_latch* si porta al valore finale dell'accumulazione nel periodo PWM.

PS: nell'altro ramo l'accumulatore viene comandato con il segnale *Sload_Negato* mentre l'*lpm_latch* viene comandato con il segnale *Sync_Diretto*.

Per la creazione del blocco *lpm_latch* seguire i seguenti passaggi. Selezionare *Edit* → *Insert Symbol*. Quindi selezionare *libraries* → *megafunctions* → *storage* → *lpm_latch*. Se si sono seguiti correttamente i passaggi elencati appare la schermata riportata in Fig. 3.26.

Premere *Ok* e nella schermata successiva scegliere il nome del blocco e premere *Next*. Nella videata che appare, alla voce *How wide should the bus be?* selezionare dal menù a tendina 14bit. In questo modo si imposta una profondità di 14bit per i bus di ingresso e di uscita dell'*lpm_latch*. Quindi premere *Finish* e posizionare il blocco *lpm_latch* sullo schematico come riportato in Fig. 3.27. Dalla quale si evince che, nel progetto, tale blocco è stato nominato *memoria*.

3.5.10 Selezione del canale di uscita

Rimane ora il problema di collegare l'uscita al ramo corretto. Per far questo bisogna selezionare, in maniera alternata, il ramo che ha di volta in volta l'accumulatore interdetto e il blocco *memoria* con l'uscita congelata al valore dell'accumulazione sul periodo. A tale scopo è stato creato il blocco *selezione*. La creazione di un blocco Verilog HDL è stata già affrontata nella sottosezione 3.5.4. Quindi ci si limita a riportare il codice Verilog HDL del blocco di selezione del canale d'uscita.

```
module selezione (in1, sync, in2, out);     %nome del modulo e definizione di ingressi
                                            %e uscite
output reg [13:0] out;                      %definizione del registro di uscita a 14 bit
input [13:0] in1;                           %definizione dell'ingresso 1 a 14bit
input [13:0] in2;                           %definizione dell'ingresso 2 a 14bit
input sync;                                 %definizione dell'ingresso sync a 1bit
```

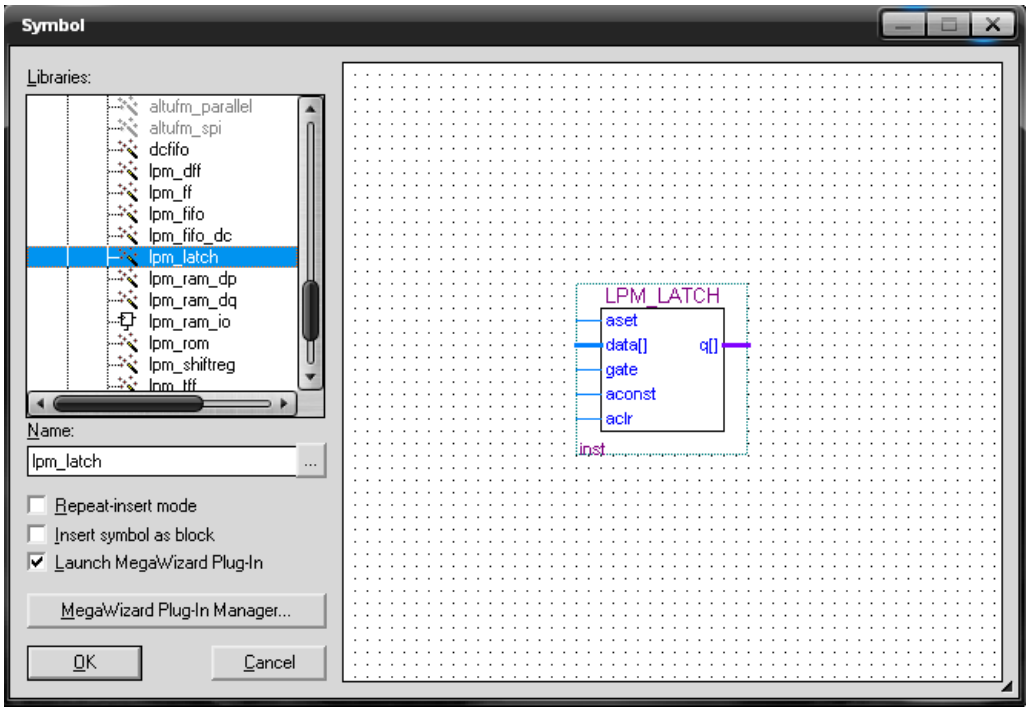

Fig. 3.26: Blocco di memorizzazione dell'accumulazione sul periodo PWM

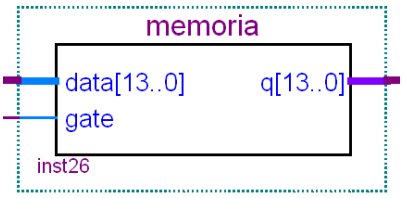

Fig. 3.27: Blocco di memorizzazione dell'accumulazione sul periodo PWM posizionato sullo schematico

```
always                                  %definizione di un ciclo infinito
begin                                   %inizio del ciclo
%se sync = 1 l'uscita viene posta uguale all'ingresso 1
if (sync)
out = in1;
%altrimenti se sync = 0 l'uscita viene posta uguale all'ingresso 2
else
out = in2;
end                                     %fine del ciclo
endmodule                               %fine del modulo
```

Nell'ingresso *in1* verrà collegata l'uscita del blocco *memoria* del canale 1; nell'ingresso *in2* verrà collegata l'uscita del blocco *memoria* del canale 2. Nell'ingresso *sync* verrà collegato il segnale *Sync_Diretto*.

In Fig. 3.28 si riporta il blocco di selezione del canale di uscita, realizzato sullo schematico.

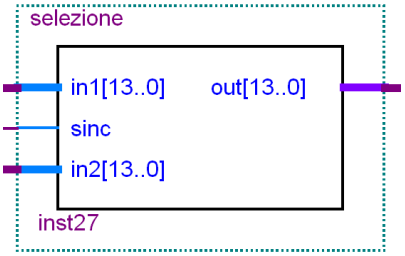

Fig. 3.28: Blocco di selezione del canale di uscita

3.5.11 Blocco di uscita e Schematico Completo

Rimane soltanto da collegare l'uscita del blocco *selezione* ai pin di ingresso del convertitore DA. Per fa questo si deve introdurre un blocco di uscita a 14bit. La procedura di creazione del blocco di uscita è del tutto analoga a quella spiegata in sottosezione 3.5.1 per il blocco di ingresso. In Fig. 3.29 viene riportato lo schematico completo, con tutti i vari blocchi, approfonditi nelle sottosezioni precedenti, collegati in cascata. Insieme al canale A di elaborazione della tensione concatenata (v-u). É stato riportato anche lo schema del canale B di elaborazione della concatenata (u-w) (identico allo schema del canale A).

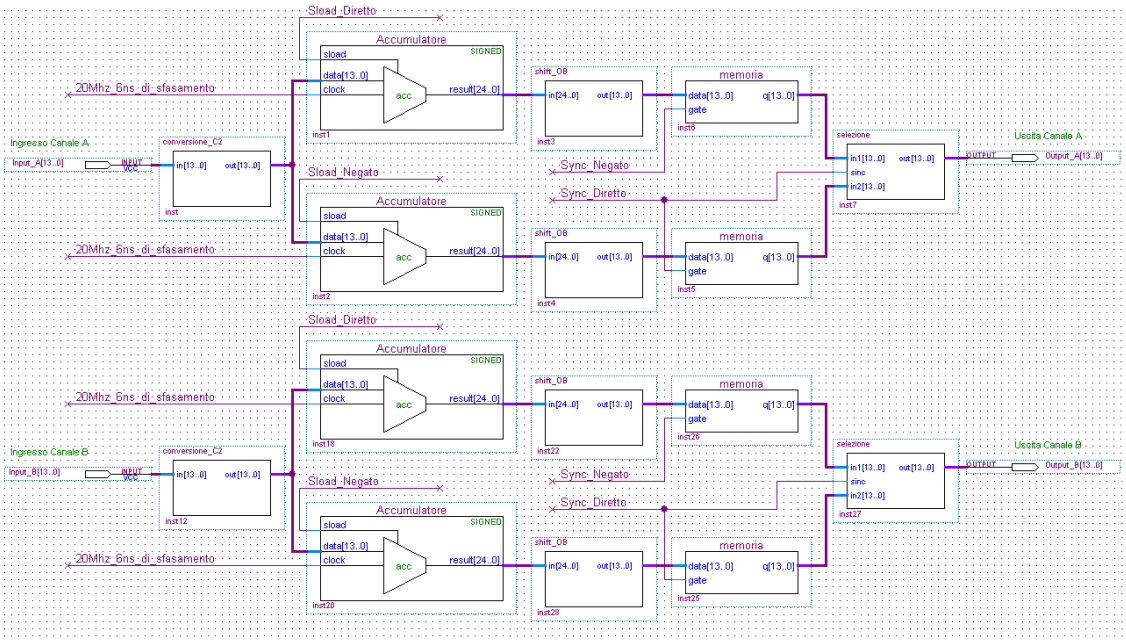

Fig. 3.29: Schematico completo con entrambi i canali di elaborazione A e B

3.5.12 Compilazione e assegnazione dei pin nell'FPGA

Una volta che il posizionamento dei blocchi è stato ultimato, bisogna assegnare ai pin di I/O utilizzati i pin fisicamente presenti nell'FPGA. Prima dell'operazione di assegnazione vera e propria bisogna compilare il programma. Ciò a due principali utilità.

- verifica la presenza di eventuali errori
- crea la lista dei pin di I/O utilizzati

Per compilare il programma basta selezionare il precorso *Processing* → *Start Compilation*.

PS: in questa tesi l'operazione di compilazione è stata riportata soltanto alla fine, ovvio sottolineare come questa fase debba essere ripetuta moltissime volte fin dall'inizio della programmazione, ai fini di eliminare fin da subito gli errori presenti. Creare un intero schematico e compilare soltanto alla fine, sarebbe del tutto deleterio a causa dell'enorme quantità di errori che sarebbero presenti e, soprattutto delle loro interazioni molto spesso di difficile discernimento.

Se la compilazione non produce errori si ottiene il seguente messagggio *Full Compilation was succesfull*; pertanto si può proseguire all'assegnazione dei pin.

Selezionando le voci *Assignments* → *Pins* verrà aperta la schermata denominata *Pin Planner* riportata in Fig. 3.30. In alto si può notare l'effettivo posizionamento dei pin all'interno dell'FPGA. Mentre in basso, la compilazione appena effettuata ha prodotto la lista dei pin di I/O utilizzati nel progetto. Le voci più importanti presenti sono il *Node Name* (cioè il nome del pin nello schematico), la Direction (cioè la direzione del pin) e la *Location* (cioè la posizione fisica del pin nell'FPGA).

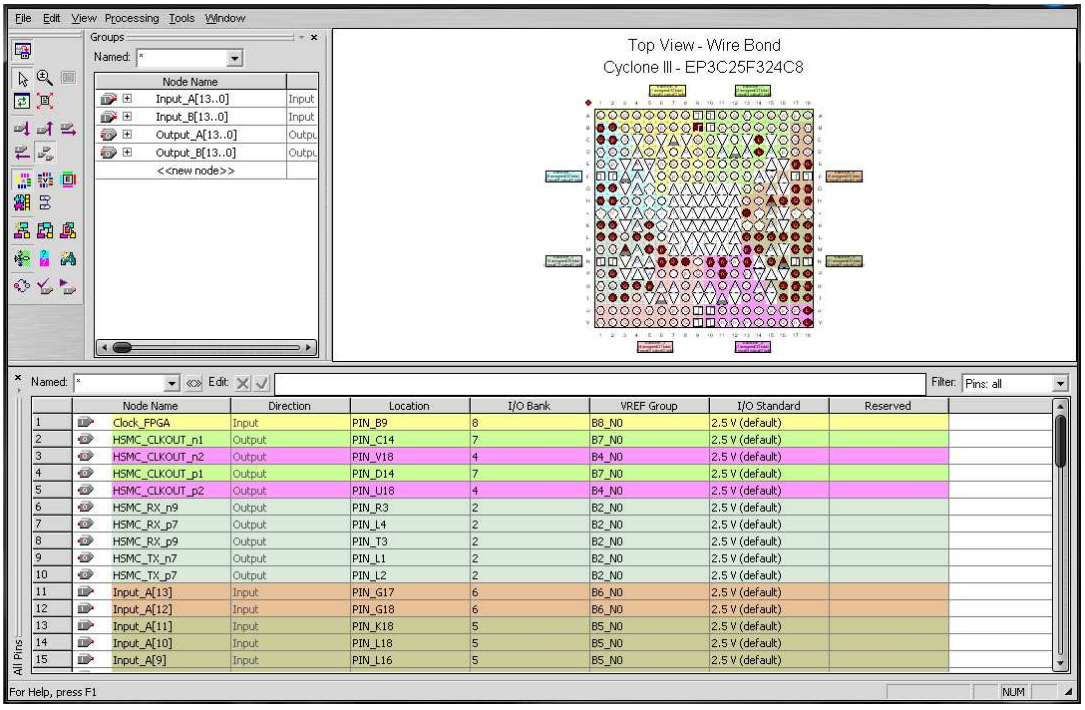

Fig. 3.30: Schermata di assegnazione dei pin dell'FPGA

Per ciascun pin dello schematico, nella casella *Location* si dovrà selezionare il nome del pin realmente presente nell'FPGA (da abbinare appunto al pin utilizzato nel programma). I nomi dei pin dell'FPGA si possono trovare in [9]. Le altre voci presenti nella lista, quali *I/O Bank*, *VREF Group*, *I/O Standard* verranno compilate automaticamente una volta effettuata la selezione dei pin alla voce *Location*. In Tab. 3.5 sono elencati gli abbinamenti dei pin di INPUT, mentre in Tab. 3.6 sono elencati gli abbinamenti dei pin di OUTPUT. Nell'ultima colonna delle tabelle è esplicitata la funzione dei pin nel programma. Sono stati riportati tutti i pin presenti nel progetto, sia quelli del canale A, che quelli del canale B.

Pin nello schematico	Pin nell'FPGA	Funzione del pin nel programma
Clock_FPGA	B9	Clock interno all'FPGA 50MHz
Sync_PWM	P2	Segnale di sincronismo con la PWM
Input_A[0]	N13	Bit 0 di ingresso del canale A
Input_A[1]	M13	Bit 1 di ingresso del canale A
Input_A[2]	N6	Bit 2 di ingresso del canale A
Input_A[3]	M6	Bit 3 di ingresso del canale A
Input_A[4]	R18	Bit 4 di ingresso del canale A
Input_A[5]	R17	Bit 5 di ingresso del canale A
Input_A[6]	M14	Bit 6 di ingresso del canale A
Input_A[7]	L13	Bit 7 di ingresso del canale A
Input_A[8]	M17	Bit 8 di ingresso del canale A
Input_A[9]	L16	Bit 9 di ingresso del canale A
Input_A[10]	L18	Bit 10 di ingresso del canale A
Input_A[11]	K18	Bit 11 di ingresso del canale A
Input_A[12]	G18	Bit 12 di ingresso del canale A
Input_A[13]	G17	Bit 13 di ingresso del canale A
Input_B[0]	T18	Bit 0 di ingresso del canale B
Input_B[1]	T17	Bit 1 di ingresso del canale B
Input_B[2]	R4	Bit 2 di ingresso del canale B
Input_B[3]	R5	Bit 3 di ingresso del canale B
Input_B[4]	P18	Bit 4 di ingresso del canale B
Input_B[5]	P17	Bit 5 di ingresso del canale B
Input_B[6]	L15	Bit 6 di ingresso del canale B
Input_B[7]	L14	Bit 7 di ingresso del canale B
Input_B[8]	M18	Bit 8 di ingresso del canale B
Input_B[9]	L17	Bit 9 di ingresso del canale B
Input_B[10]	H18	Bit 10 di ingresso del canale B
Input_B[11]	H17	Bit 11 di ingresso del canale B
Input_B[12]	E18	Bit 12 di ingresso del canale B
Input_B[13]	E17	Bit 13 di ingresso del canale B

Tab. 3.5: Assegnazione dei pin di input e loro funzione nel programma

Pin nello schematico	Pin nell'FPGA	Funzione del pin nel programma
HSMC_CLKOUT_n2	V18	Clock per l'ADC canale A
HSMC_CLKOUT_p2	U18	Clock per l'ADC canale B
HSMC_CLKOUT_n1	C14	Clock per il DAC canale A (Latch DAC_A)
HSMC_CLKOUT_p1	D14	Clock per il DAC canale B (Latch DAC_B)
HSMC_TX_p[7]	L2	Clock per il DAC canale A (Latch di ingresso A)
HSMC_RX_p[7]	L4	Clock per il DAC canale B (Latch di ingresso B)
HSMC_RX_p[9]	T3	"0" ⇒ canale A di uscita dell'ADC attivo
HSMC_Rx_n[9]	R3	"0" ⇒ canale B di uscita dell'ADC attivo
HSMC_TX_n[7]	L1	"1" ⇒ modalità "dual port" del DAC attivata
Output_A[0]	M5	Bit 0 di uscita del canale A
Output_A[1]	N7	Bit 1 di uscita del canale A
Output_A[2]	N8	Bit 2 di uscita del canale A
Output_A[3]	J13	Bit 3 di uscita del canale A
Output_A[4]	N10	Bit 4 di uscita del canale A
Output_A[5]	N11	Bit 5 di uscita del canale A
Output_A[6]	K17	Bit 6 di uscita del canale A
Output_A[7]	P11	Bit 7 di uscita del canale A
Output_A[8]	B2	Bit 8 di uscita del canale A
Output_A[9]	B1	Bit 9 di uscita del canale A
Output_A[10]	G2	Bit 10 di uscita del canale A
Output_A[11]	G1	Bit 11 di uscita del canale A
Output_A[12]	K2	Bit 12 di uscita del canale A
Output_A[13]	K1	Bit 13 di uscita del canale A
Output_B[0]	M3	Bit 0 di uscita del canale B
Output_B[1]	T2	Bit 1 di uscita del canale B
Output_B[2]	H15	Bit 2 di uscita del canale B
Output_B[3]	H16	Bit 3 di uscita del canale B
Output_B[4]	N16	Bit 4 di uscita del canale B
Output_B[5]	N15	Bit 5 di uscita del canale B
Output_B[6]	R16	Bit 6 di uscita del canale B
Output_B[7]	T16	Bit 7 di uscita del canale B
Output_B[8]	C2	Bit 8 di uscita del canale B
Output_B[9]	C1	Bit 9 di uscita del canale B
Output_B[10]	H2	Bit 10 di uscita del canale B
Output_B[11]	H1	Bit 11 di uscita del canale B
Output_B[12]	K5	Bit 12 di uscita del canale B
Output_B[13]	L5	Bit 13 di uscita del canale B

Tab. 3.6: Assegnazione dei pin di output e loro funzione nel programma

Una volta assegnati i pin, ricompilare il programma, e verificare che non vi siano errori. Durante questa fase si è verificato il seguente problema. La compilazione da un errore sul pin `E18`, che corrisponde al pin 12 di ingresso del canale B dell'ADC. Nello specifico viene suggerito dal compilatore che tale pin risulta già assegnato ad una funzione interna all'FPGA denominata *nCEO*. Per ovviare a tale problema, e permettere al programma di utilizzare tale pin come un nomale pin di I/O basta seguire i seguenti passaggi. Selezionare il percorso *Assignments* → *Device*. Compare una schermata con dei menù a sinistra selezionabili. Cliccare su *Device* e nella videata che compare selezionare *Device and Pin Options*. A questo punto visualizzare il menù *Dual-Purpose Pins* e selezionare dal menù a tendina di fianco alla voce *nCEO* la scelta *Use as regular I/O*; che, come si intuisce dal nome, permette di utilizzare il pin `E18` come un normale pin di I/O.

La compilazione oltre alla descrizione di eventuali errori nel programma produce un riassunto delle risorse hardware utilizzate dal programma stesso. Tale riassunto è definito *Compilation Report* ed è riportato in Fig. 3.31.

Flow Status	Successful - Mon Oct 04 20:25:40 2010
Quartus II Version	9.1 Build 304 01/25/2010 SP 1 SJ Web Edition
Revision Name	tesi
Top-level Entity Name	tesi
Family	Cyclone III
Device	EP3C25F324C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	265 / 24,624 (1 %)
Total combinational functions	263 / 24,624 (1 %)
Dedicated logic registers	119 / 24,624 (< 1 %)
Total registers	119
Total pins	67 / 216 (31 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)

Fig. 3.31: Riassunto delle risorse hardware utilizzate dal programma

Infine in Fig. 3.32 si possono osservare entrambi i canali di elaborazione compresi dei pin di ingresso e uscita assegnati.

3.5.13 Caricamento del programma nell'FPGA

Prima di procedere, si devono installare nel Personal Computer i driver dell'`USB Blaster`, che è il circuito presente all'interno della `Cyclone_III_Starter_Board` utilizzato per caricare il programma dal PC all'FPGA, attraverso il cavo USB fornito in dotazione con la scheda. Di seguito vengono riportate le istruzioni per l'installazione dei driver, trovate in [14]. Essendo la spiegazione ben organizzata e molto facile da capire, viene lascita in inglese.

1. Before you begin the installation, verify the `USB-Blaster driver` is located in your directory: \<*Quartus II system directory*>*drivers**usb-blaster*
2. If the driver is not in your directory, download the `USB-Blaster driver` from the Altera web site [15]
3. Plug in the `USB-Blaster download` cable to the PC
4. On the *Found New Hardware Wizard* window, click *No, not this time* and then click *Next* to continue
5. Select *Install from a list of specific location (Advanced)* and click *Next* to continue
6. Select *Don't search. I will choose the driver to install.* Click *Next*

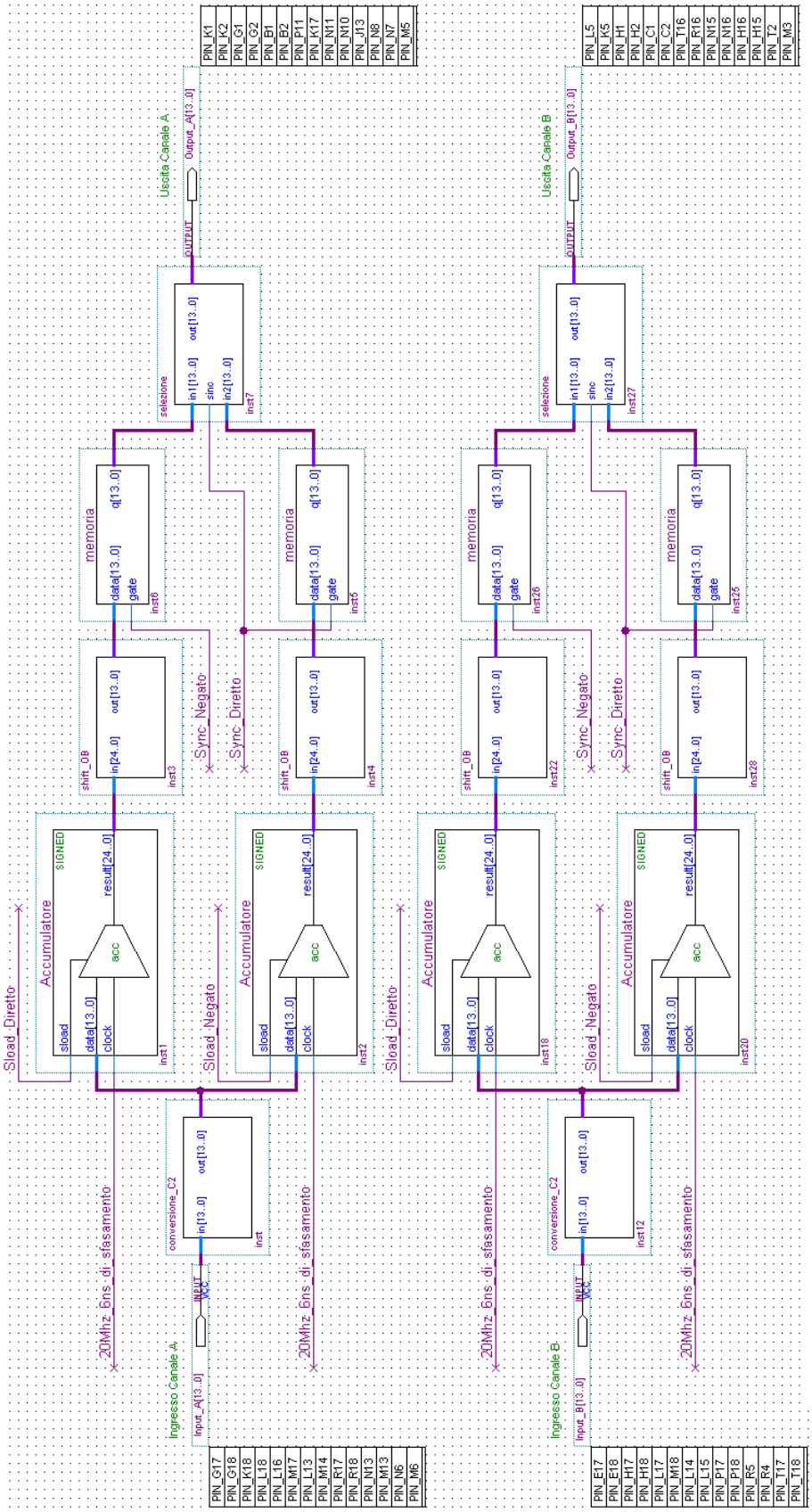

Fig. 3.32: Canali di elaborazione A e B con pin di ingresso e uscita assegnati

7. Select *Sound, video and game controllers*, and click *Next* to continue
8. Select *Have Disk* and browse to the location of the driver on your system: \<*Quartus II system directory*>*drivers**usb-blaster*. Click *OK*
9. Select `Altera USB-Blaster` and click *Next* to continue
10. Click *Next* to install the driver
11. Click *Continue Anyway* when the *Hardware Installation warning appears*
12. Click *Finish* in the *Completing the Add/Remove Hardware Wizard* window and reboot your system

Ora che i driver dell'`USB Blaster` sono installati si può procedere.

La compilazione precedentemente eseguita ha prodotto un file con estensione *.sof* (SRAM Object File). Tale file viene utilizzato nella procedura di caricamento del programma nell'FPGA descritta di seguito. Preliminarmente, se non lo si è già fatto, accendere la scheda e collegarla al PC tramite il cavo USB. Ora selezionare il percorso *Tools* → *Programmer*. In questo modo si aprirà la schermata in Fig. 3.33.

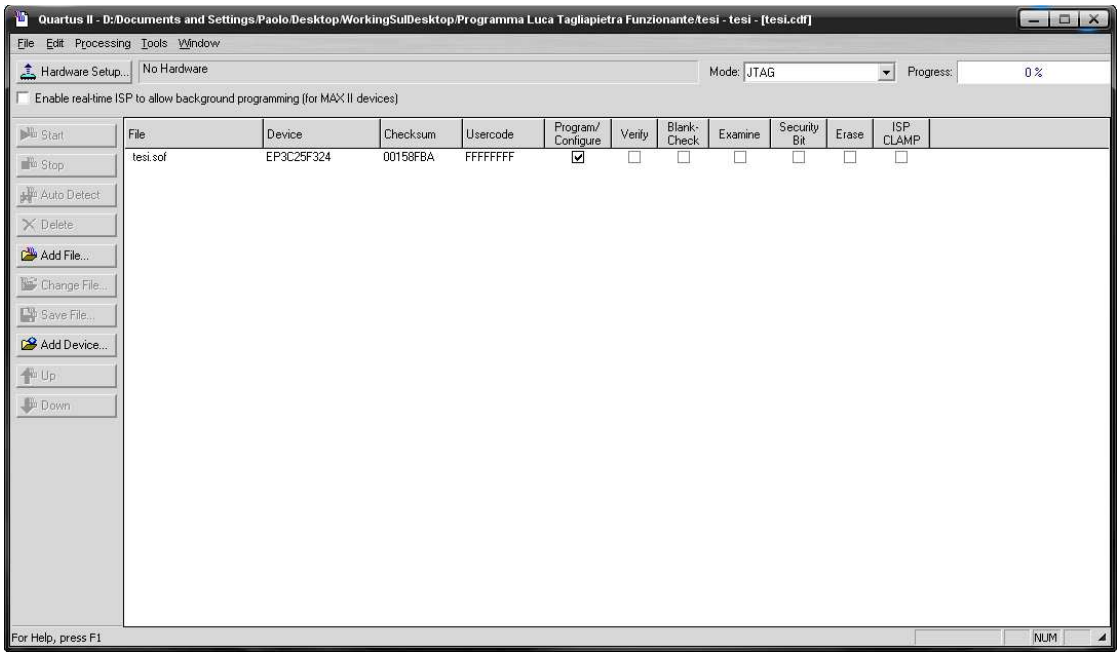

Fig. 3.33: Schermata per il caricamento del software nell'FPGA

Cliccando la voce in alto a sinistra *Hardware Setup* appare la schermata di selezione della modalità di connessione alla scheda. Nel menù a tendina di fianco alla voce *Currently selected hardware* scegliere l'opzione *USB-Blaster [USB-0]*, cliccare il pulsante *Add hardware* e premere *Close*. Successivamente scegliere la voce *Add File* e, nella cartella del progetto, selezionare il file con estensione *.sof* e premere *Open*. A questo punto il nome del file è visibile nella schermata di Fig. 3.33. Ora basta semplicemente spuntare la casella *Program/Configure* (se non dovesse essere già selezionata) e premere il pulsante *Start*. Quanto la barra *Progress* raggiunge il 100% l'FPGA è programmata correttamente, e quindi si può passare alla fase di test dell'Algoritmo.

Capitolo 4

SIMULAZIONI

4.1 Simulator Tool

Nella fase di progettazione e debugging del software è stato utilizzato il *Simulator Tool* fornito dal *Quartus II*. É uno strumento molto utile ai fini di valutare l'analisi funzionale dei vari blocchi dello schematico. Permette al programmatore di comprendere e correggere l'eventuale mal funzionamento del codice programmato. Le videate riportate nelle figure di sezione 4.1 e sezione 4.2 (ove non specificato altrimenti) sono state estrapolate dal `Quartus II`.

Il *Simulator Tool* è in grado di fornire a ciascun blocco di ingresso, presente nello schematico, dei segnali a scelta, e di valutare il comportamento dei blocchi di uscita. É caratterizzato dalla possibilità di utilizzare due diversi tipi di analisi, funzionale e temporale. Nel progetto è stata utilizzata soltanto l'analisi funzionale, che per gli scopi affrontati è in pratica coincisa con l'analisi temporale. La quale non è stata utilizzata in quanto ci è sembrata non particolarmente utile. Di seguito si cerca di spiegare meglio quanto appena affermato.

Nell'esempio di analisi che verrà decritto a breve, a causa della tipologia particolare del problema affrontato, anche se si è utilizzata solamente l'analisi funzionale, è intrinsecamente presente anche un risultato che considera la tempistica dei blocchi simulati. Infatti in questo caso l'analisi funzionale e quella temporale forniscono il medesimo risultato.

L'analisi temporale sarebbe stata utile a comprendere l'effettivo tempo di elaborazione dei dati, impiegato dai blocchi presenti in libreria (come potrebbero essere delle semplici porte NOT in cascata). Purtroppo la simulazione temporale in questo caso non ha mai dato dei risultati soddisfacenti e quindi non è stata utilizzata. In particolare il risultato dell'analisi temporale applicata ad una semplice porta NOT, restituisce un tempo di propagazione del segnale, all'interno della porta stessa, pari a 6ns. Aggiungendo altre porte uguali in cascata il tempo rimane costante. Tale risultato è inverosimile, in quanto ci si aspetterebbe un tempo circa uguale alla moltiplicazione dei 6ns per il numero di porte NOT utilizzate. Per cui l'analisi temporale non fornisce utili informazioni circa il tempo di propagazione dei segnali all'interno dei blocchi realizzati nello schematico.

Il *Simulator Tool* è stato particolarmente utile nella simulazione del circuito di creazione del ritardo, approfondito in sottosezione 3.5.4 (alla quale si rimanda fortemente prima di leggere la parte seguente). Tale circuito si propone di ottenere un segnale *Sload* di uscita, caratterizzato dagli stessi fronti di discesa del segnale *Sync* di ingresso, ma con fronti di salità ritardati di 12,8μs (ritardo ricavato dal calcolo teorico).

Come spiegato in sottosezione 3.5.4, i circuti ritardatori sono due e il loro funzionamento è duale. Quindi in questa sezione verrà simulato solamente il circuito di condizionamento del segnale *Sync_Diretto*. I cui risultati sono validi anche per il circuito di condizionamento del segnale *Sync_Negato*.

Prima di procedere alla creazione del file di simulazione si devono introdurre nello schematico dei blocchi di uscita nei punti in cui si desidera monitorare il segnale processato. In particolare si desidera osservare i seguenti punti dello schematico:

- il nodo *Sync_Diretto*
- il nodo *Sload_Diretto*
- il nodo di uscita del contatore

Per i quali sono stati creati i rispettivi blocchi di uscita:

- *Pin_Sync_Diretto*
- *Pin_Sload_Diretto*
- *contatore* (che è un blocco di uscita a 9bit)

I quali sono riportati in Fig. 4.1.

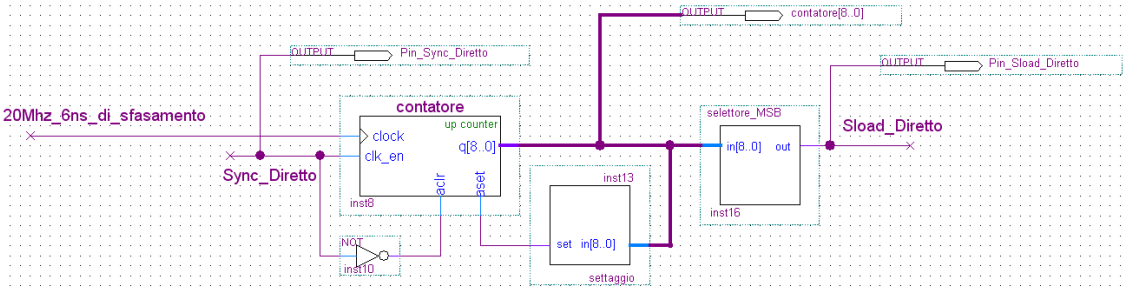

Fig. 4.1: Simulator Tool: Creazione dei blocchi di uscita

Per far funzionare il circuito di creazione del ritardo occorrono anche i seguenti ingressi:

- *Clock_FPGA*
- *Sync_PWM*

I quali non hanno bisongno della creazione di ulteriori blocchi di ingresso (essendo appunto essi già dei blocchi di ingresso).

Ora compilare il progetto con il percorso *Processing* → *Start Compilation*. Se non ci sono errori, si può iniziare la fase di simulazione vera e propria.

Selezionare *File* → *New*, nella finestra che compare scegliere *Vector Waveform File* e premere *OK*. Comparirà la finestra di gestione della simulazione. Si consiglia di cliccare col tasto destro in alto sull'etichetta della finestra, e premere *Detach Window*. In questo modo si può ottenere la finestra a pieno schermo riportata in Fig. 4.2.

Ora salvare il file, selezionando *File* → *Save As*, nella finestra che compare scegliere il nome del file con estensione *".vwf"* e premere *Save*.

A questo punto si possono inserire i blocchi di ingresso e uscita che interesseranno la simulazione. Per far questo effettuare un doppio click, all'interno della colonna bianca di sinistra sotto l'etichetta *Name*. Così facendo comparirà la finestra *Insert Node or Bus*. Cliccare sul pulsante *Node Finder*. In questo modo comparirà la finestra *Node Finder* riportata in Fig. 4.3.

Nel menù a tendina corrispondente al campo *Filter* selezionare la voce *Pins: all* e premere *List*. Così facendo nella colonna di sinistra compariranno tutti i pin disponibili per la simulazione. Per sceglierli basterà evidenziarli col mouse e cliccare sul pulsante ">". In questo modo i pin saranno copiati nella colonna di destra. A questo punto premere *OK* due volte, e i pin saranno ora disponibili sulla schermata di gestione della simulazione.

Si consiglia inizialmente di impostare la durata della finestra temporale da simulare; attraverso il percorso *Edit* → *End Time*. Nell'esempio in questione sono stati simulati 600μs.

Con la barra degli strumenti a sinistra si possono comandare a piacimento i segnali di ingresso. In particolare, nella simulazione effettuata, sono stati impostati gli ingressi come segue:

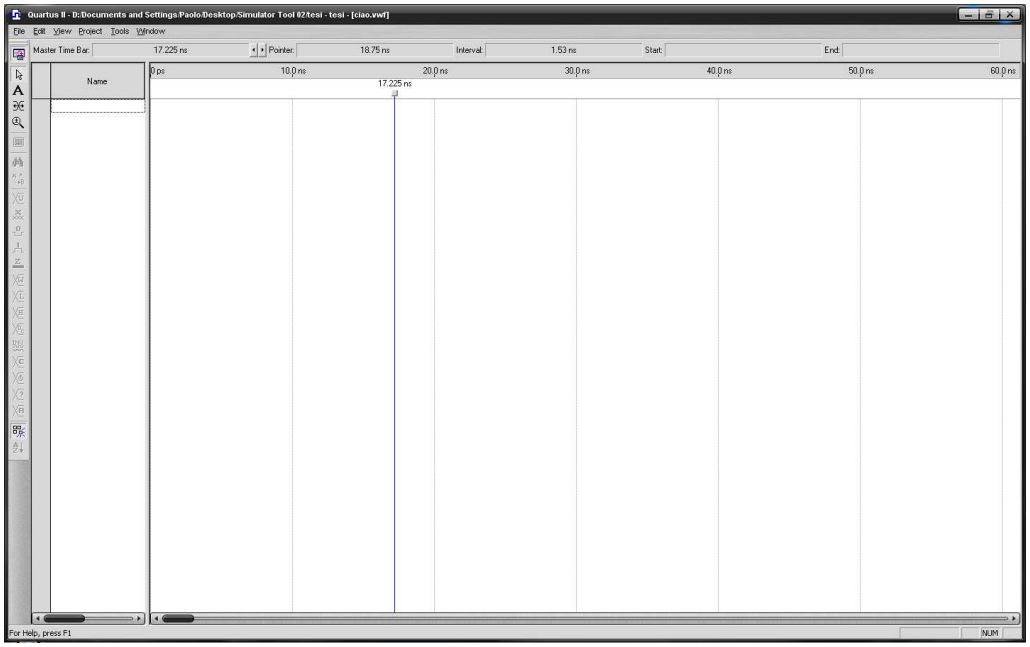

Fig. 4.2: Simulator Tool: Finestra di gestione della simulazione

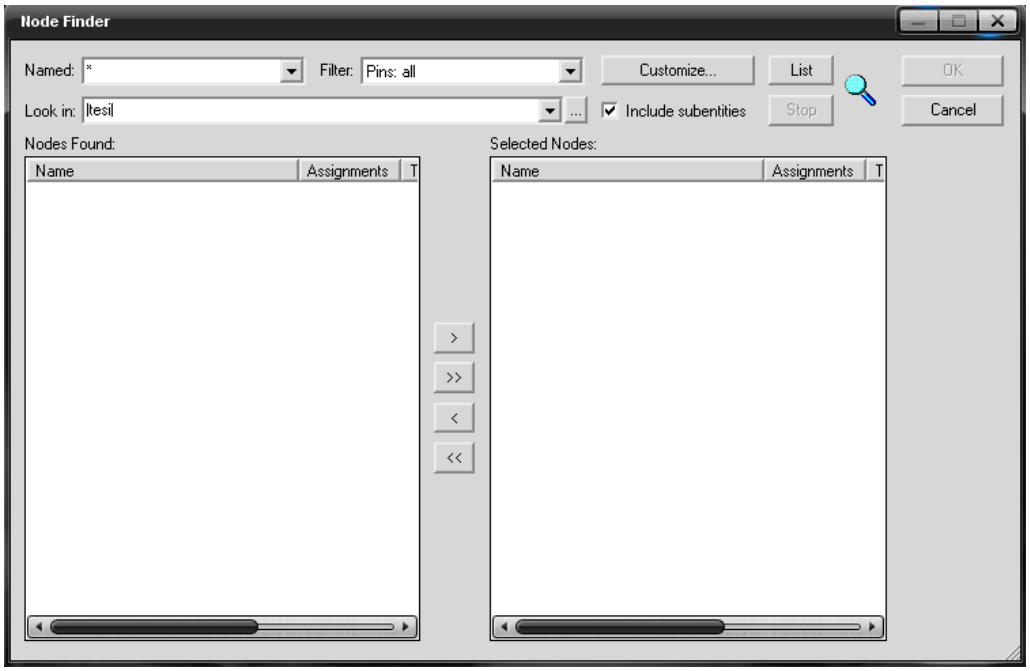

Fig. 4.3: Simulator Tool: Node Finder

- *Clock_FPGA* → Clock a 50Mhz
- *Sync_PWM* → segnale con periodo pari a 100μs e duty cycle del 40%.

Le scelte adottate vogliono simulare la situazione reale in cui il circuito di creazione del ritardo si trova ad operare.

Ovvio ricordare che ai blocchi *Pin_Sync_Diretto*, Pin_Sload_Diretto e *contatore*, non debbano essere assegnati segnali in quanto sono blocchi di uscita. Gli eventuali segnali assegnati sarebbero sovrascritti in simulazione.

Ora è necessario impostare la simulazione, per far questo selezionare, dal menù principale del `Quartus II`, il percorso *Processing* → *Simulator Tool*. Seguendo questi passaggi compare la finestra riportata in Fig. 4.4, nella quale alla voce *Simuation Mode* si deve scegliere l'opzione *Functional*, per effettuare un'analisi funzionale; mentre alla voce *Simulation Input* si deve riportare il nome del file con estensione *''.vwf''* creato precedentemente. A questo punto si può chiudere la finestra.

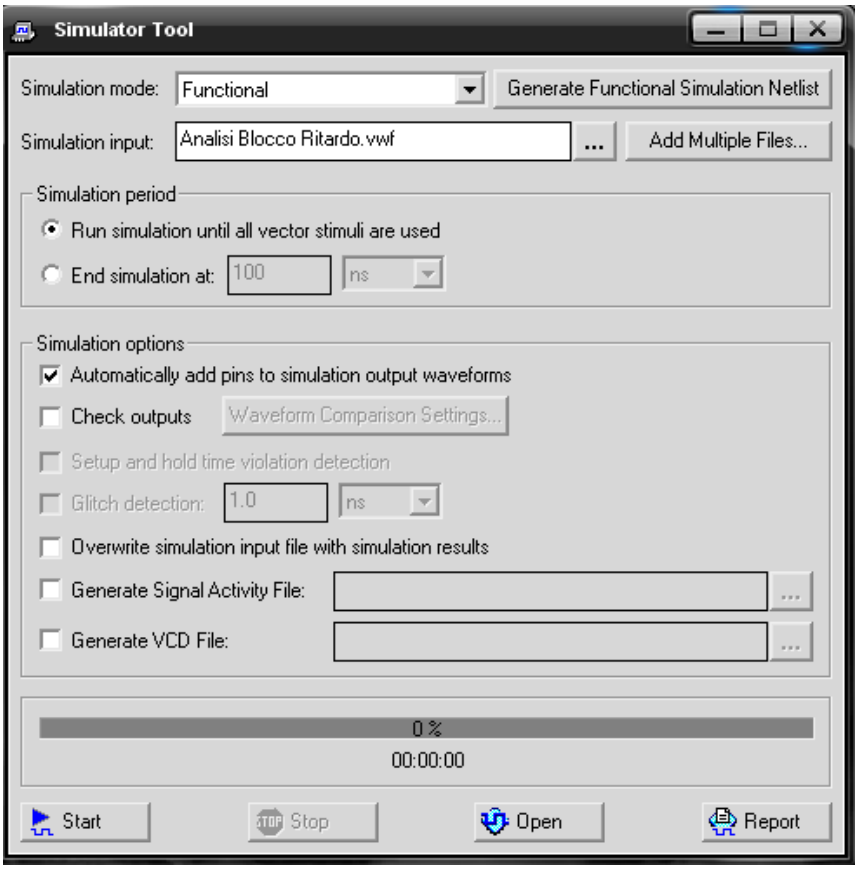

Fig. 4.4: Simulator Tool: Impostazione della simulazione

Infine bisogna ricompilare il programma utilizzando il percorso *Processing* → *Start Compilation*. Successivamente, selezionando le voci *Processing* → *Generate Functional Simulation Netlist*, si creano gli abbinamenti tra i pin di ingresso e uscita del programma e le modifiche apportate a tali pin nella schermata di gestione della simulazione.

Ora è sufficiente far partire la simulazione selezionando *Processing* → *Start Simulation*.

La simulazione del circutio di creazione del ritardo è riportata in Fig. 4.5, nella quale possiamo notare il funzionamento specifico dei vari blocchi in esame. Il primo segnale a partire dall'alto è il clock dell'FPGA. É una banda nera in quanto ha una frequenza (pari a 50Mhz) molto superiore ai segnali di interesse in questo esempio. Il secondo segnale è il

segnale di sincronismo della PWM caratterizzato da un periodo pari a 100μs (10Khz) e un duty cycle del 40%. Il terzo segnale è il sincronismo diretto realizzato a partire dal segnale di sincronismo PWM attraverso l'utilizzo di un Flip Flop edge triggered (vedi sottosezione 3.5.4). Come ci si aspettava, il segnale è caratterizzato da un periodo doppio (rispetto al sincronismo della PWM) pari a 200ns (5Khz) e un duty cycle del 50%. Si fa infine notare come il quarto segnale, denominato nello schematico *Sload_Diretto*, coincida con il MSB dell'uscita del contatore. Quindi, come si desiderava ottenere, rispetto al segnale *Sync_Diretto* è solo il fronte di salita che viene ritardato, mentre il fronte di discesa rimane invariato.

Il ritardo coincide con il tempo impiegato dal contatore per portare a uno il MSB, che nell'analisi teorica effettuata in sottosezione 3.5.4, dovrebbe coincidere con 12,8μs. Nella Fig. 4.6 si può notare un ingrandimento della parte relativa al conteggio del contatore. Sfruttando le barre temporali fornite dalla simulazione, si può osservare come la simulazione stessa indichi un ritardo pari a 12,78μs, perfettamente aderente al calcolo teorico.

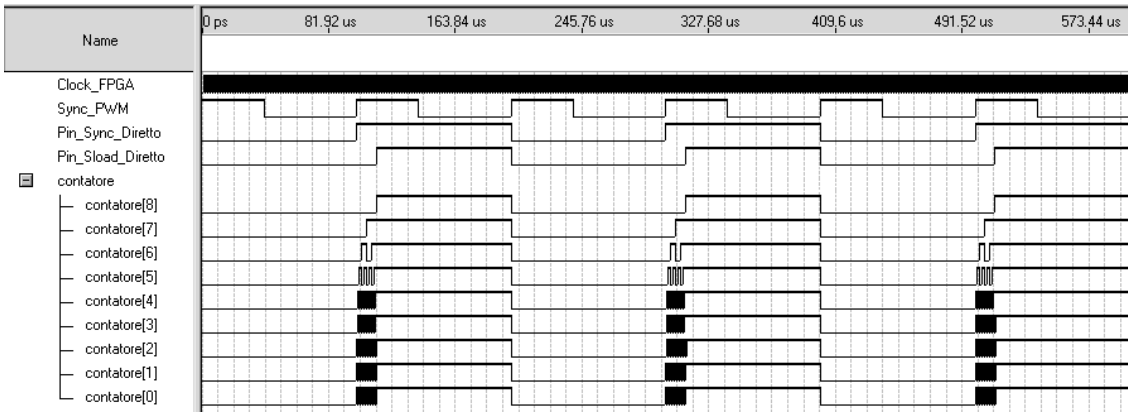

Fig. 4.5: Simulator Tool: Analisi Blocco Ritardo

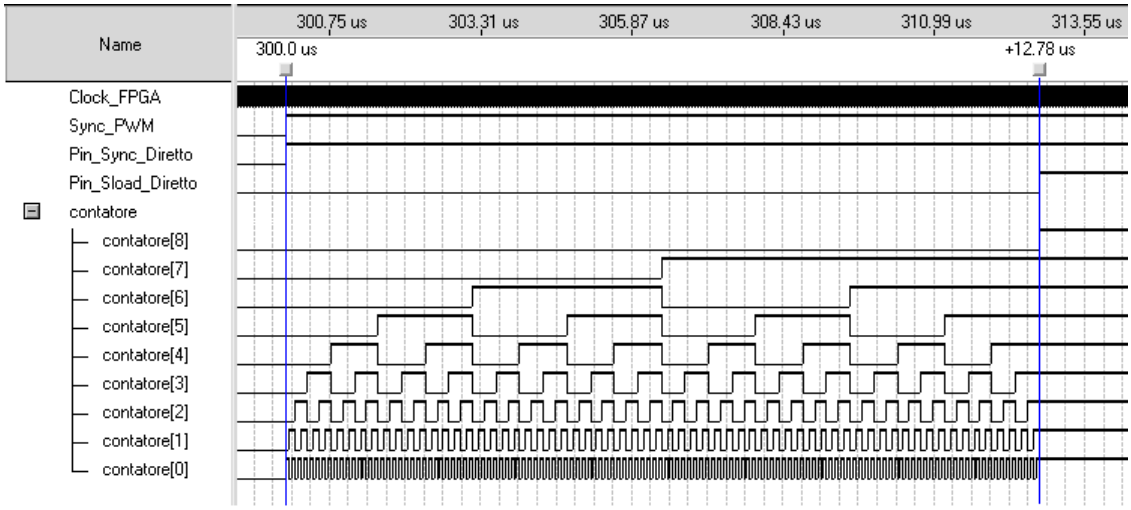

Fig. 4.6: Simulator Tool: Analisi Blocco Ritardo (ingrandimento)

4.2 Signal Tap II Logic Analyzer

Prima di procedere alle misure sperimentali, si è preferito osservare il corretto funzionamento del software utilizzando un tool fornito dal `Quartus II` stesso. Si tratta di un oscilloscopio virtuale, che permette di osservare nel PC il comportamento del programma, a fronte di opportuni segnali di ingresso forniti al sistema di misura. Tale strumento viene denominato `Signal Tap II Logic Analyzer`, ed in particolare consente all'utente di osservare il comportamento di tutti i blocchi di ingresso e uscita presenti nel programma. Il `Signal Tap II Logic Analyzer` non è stato molto sfruttato, in quanto si è preferito utilizzare l'oscilloscopio vero e proprio. Ma è stato utile in una prima fase di debugging del software, per verificare se i vari blocchi si comportassero in maniera corretta. Di seguito varrà illustrato come implementare l'oscilloscopio virtuale.

Dopo aver assegnato i pin di ingresso e uscita, e aver verificato che la compilazione sia andata a buon fine; selezionare il percorso *File* → *New*, nella finestra che compare scegliere la voce *Signal Tap II Logic Analyzer File* e premere *OK*. Così facendo compare la finestra visualizzata in Fig. 4.7.

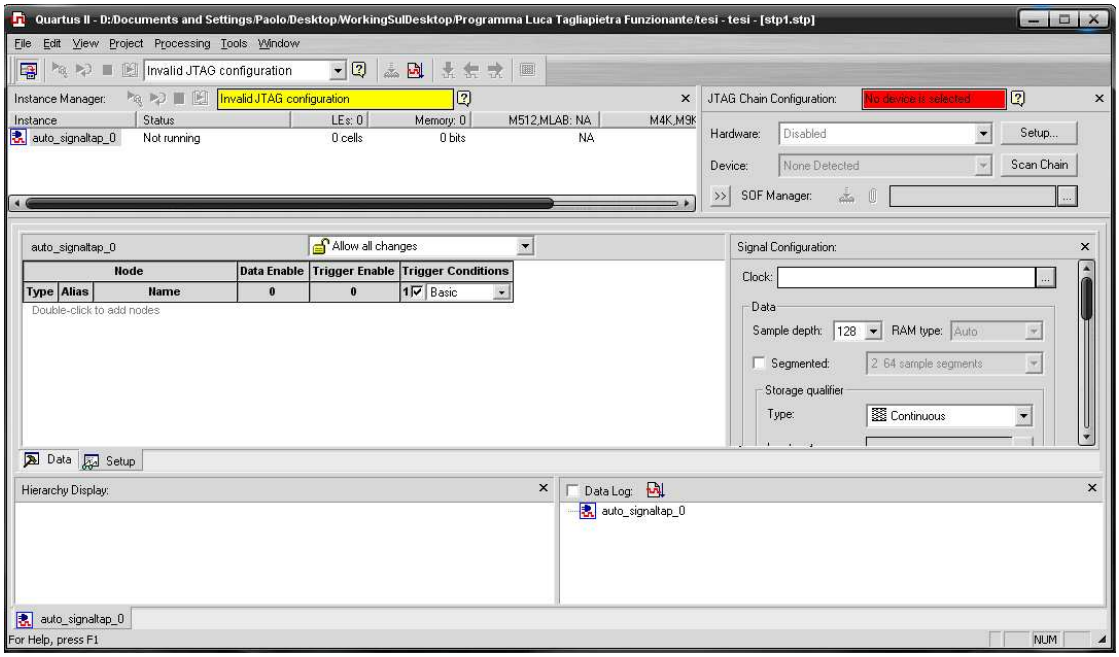

Fig. 4.7: SignalTap II Logic Analyzer: Schermata principale

Selezionare *File* → *Save As* e, nella directory del progetto, scegliere il nome del file con estensione *.stp*, premere *Save*. Alla domanda *Do you want to enable SignalTap II File ''nome_file.stp'' for the current project?* rispondere *Yes*, in questo modo si è realizzato un *SignalTap File* per il progetto corrente.

A questo punto è necessario configurare la parte hardware. Per far questo, in alto a destra, premere il pulsante *Setup*; e, nella finestra che compare, nel menù a tendina di fianco alla voce *Currently selected hardware:* selezionare *USB-Blaster [USB-0]*. In questo modo il *Signal Tap* apprende che il collegamento con la scheda FPGA è effettuato attraverso il circuito `USB-Blaster`, i cui driver sono stati installati in sottosezione 3.5.13. É appunto attraverso il cavo USB, che l'oscilloscopio virtuale leggerà il comportamento dei nodi da monitorare, la scelta dei quali viene spiegata di seguito.

Per introdurre nel *Signal Tap* i nodi da visualizzare sull'oscilloscopio virtuale, selezionare la finestra con l'etichetta *Data* in basso a sinistra; successivamente effettuare

un doppio click sullo spazio bianco al centro della pagina, come suggerito dalla scritta *Double-click to add nodes*. In questo modo compare la finestra riportata in Fig. 4.8.

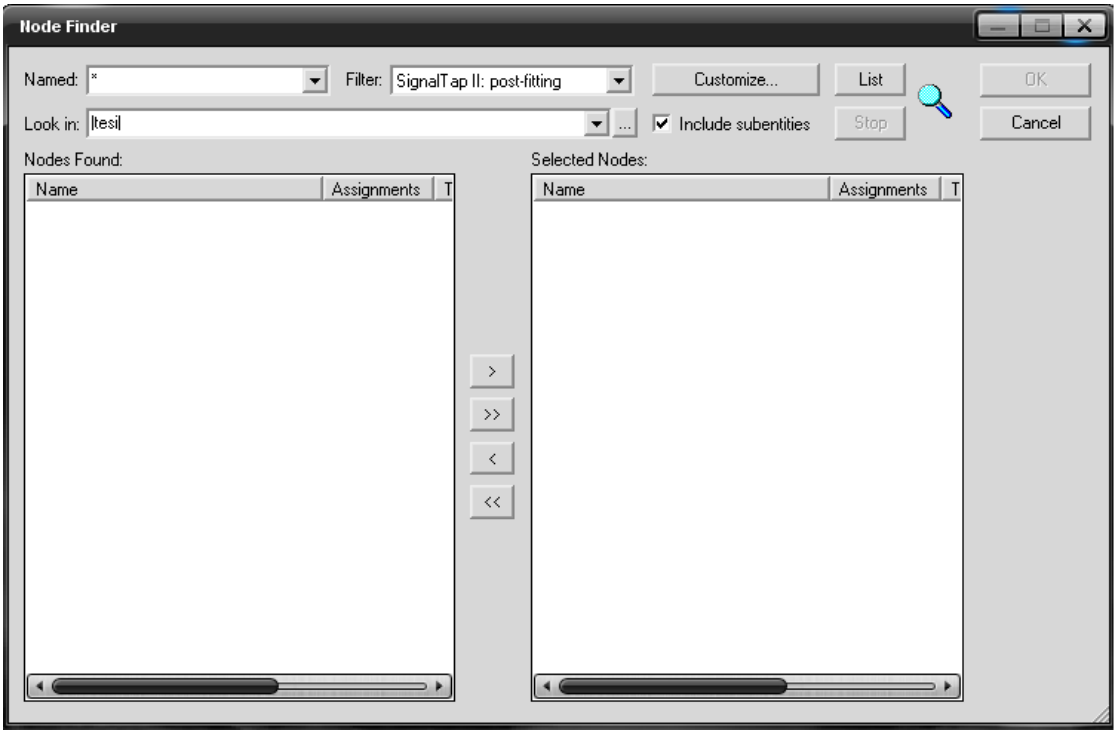

Fig. 4.8: SignalTap II Logic Analyzer: Selezione dei nodi da monitorare

Per creare la lista dei nodi effettivamente visualizzabili nell'oscilloscopio virtuale, selezionare nel menù a tendina, di fianco alla voce *Filter:* l'opzione *SignalTap II: pre-synthesis* e premere il pulsante *List*. In questo modo compare sulla colonna di sinistra la lista dei nodi selezionabili. Per sceglierli basta evidenziarli col mouse e premere il pulsante "$>$". Infine è sufficiente premere *OK*. Nell'esempio che verrà a breve riportato sono stati utilizzati i nodi *Input_A*, *Output_A* e *Output_B*.

Resta ora da impostare la finestra di osservazione, con due operazioni principali:

- impostare la frequenza di campionamento del SignalTap
- impostare il numero di campioni da visualizzare nell'scilloscopio virtuale

Nel progetto è stato creato un blocchetto di uscita apposito (riportato in Fig. 4.9) che riceve dal PLL il clock da fornire al *SignalTap*.

Fig. 4.9: SignalTap II Logic Analyzer: Clock di Campionamento

Pertanto selezionare la finestra con l'etichetta *Setup* in basso a sinistra. Nella sottofinestra in alto a destra denominata *Signal Configuration*, alla voce *Clock* digitare il nome del nodo che contiene il clock da fornire al SignalTap. Nel progetto, come evidenziato in Fig. 4.9 il nome scelto è *Clock_SignalTap*. Infine, alla voce *Sample depth* si decide la quantità di campioni da visualizzare nella finestra dell'oscilloscopio. Nell'esempio riportato a breve sono stati scelti 4096 campioni.

Per rendere effettive le modifiche apportate non resta che ricompilare il progetto e caricare nuovamente il programma nell'FPGA, come spiegato in sottosezione 3.5.13.

Per avviare l'oscilloscopio virtuale è sufficiente selezionare dal menù principale la voce *Processing* e successivamente selezionare *Autorun Analysis* se si desidera visualizzare il comportamento del programma in real time; oppure selezionare *Run Analysis* se si desidera visualizzare una singola schermata di acquisizione.

Per verificare il funzionamento del programma sono stati iniettati due segnali in ingresso alla scheda di interfaccia:

- il segnale in ingresso allo stadio di condizionamento del segnale di sincronismo (vedi sottosezione 2.3.2) ha le seguenti caratteristiche: onda quadra con frequenza pari a 12Khz, valore logico basso 0V, valore logico alto 5V, duty cycle a piacere,
- il segnale in ingresso allo stadio di trasformazione da Single Ended Mode a Differential Mode (vedi sottosezione 2.3.1) ha le seguenti caratteristiche: onda sinusoidale con frequenza pari a 12Hz, Ampiezza 3Vpp, Offset 0V.

I risultati di uscita sono stati verificati sia con il SignalTap che con l'oscilloscopio a disposizione in laboratorio. In particolare ci si aspetta che l'uscita del sistema segua l'andamento sinusoidale del segnale in ingresso. In Fig. 4.10 è riportata l'elaborazione del SignalTap. La prima forma d'onda corrisponde al segnale di ingresso. La seconda forma d'onda visualizza la sommatoria prodotta da un accumulatore. Mentre la terza la forma d'onda rappresenta l'uscita del programma, che, come ci si aspettava, segue bene l'ingresso.

In Fig. 4.11 è visualizzata l'elaborazione col Matlab di un milione di campioni prelevati dall'oscilloscopio. In particolare il segnale in blu corrisponde all'operazione di sommatoria di un accumulatore. Mentre il segnale in rosso corrisponde all'uscita del programma.

Da tale figura si può notare una cosa interessante. Infatti si osserva che l'uscita non segue perfettamente il valore alto delle accumulazioni. Il motivo di tale comportamento, che qui riportiamo brevemente, è stato esaurientemente spiegato in sottosezione 3.5.4.

Il programma è costruito in modo che il valore finale dell'accumulazione venga salvato leggermente prima che l'accumulatore si resetti. In questo modo alla fine di ogni periodo PWM, l'uscita dell'accumulatore avrà un valore più elevato, (in modulo) rispetto al corretto valore di uscita del programma.

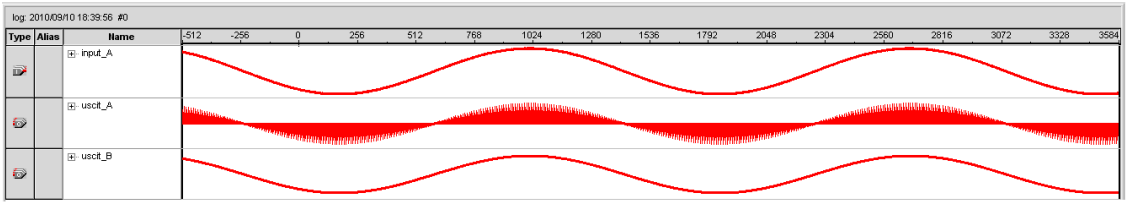

Fig. 4.10: Onda sinusoidale 12Hz: SignalTap

Infine sfruttando il SignalTap si può osservare, come riportato in Fig. 4.12, l'andamento dei singoli bit d'uscita dell'accumulatore. Si fa notare come, in maniera concorde con la codifica Offset Binary, il MSB, corrispondente al bit più in basso in Fig. 4.12, assuma il valore zero in corrispondenza di un'uscita negativa dell'accumulatore e, viceversa, assuma il valore uno in corrispondenza di un'uscita positiva dell'accumulatore.

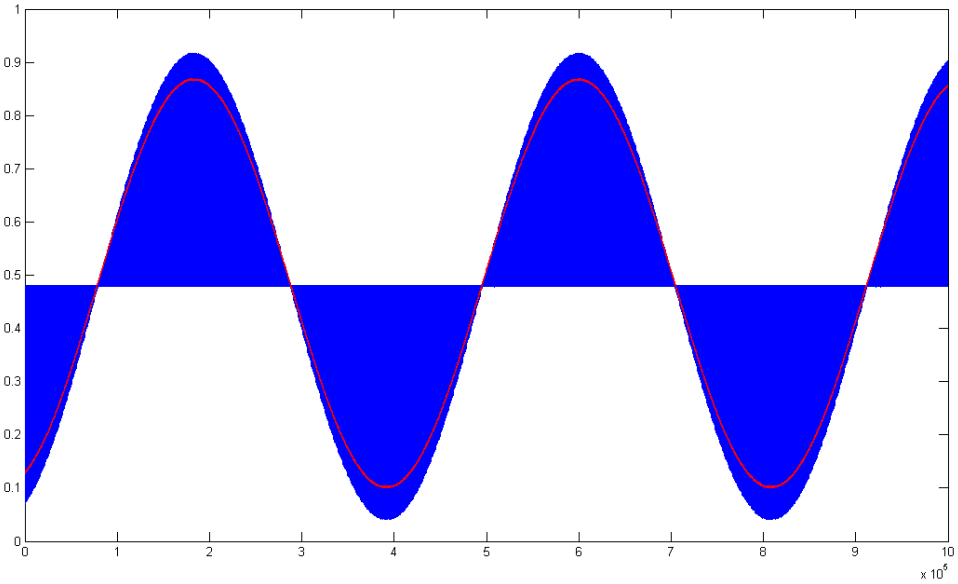

Fig. 4.11: Onda sinusoidale 12Hz: Oscilloscopio

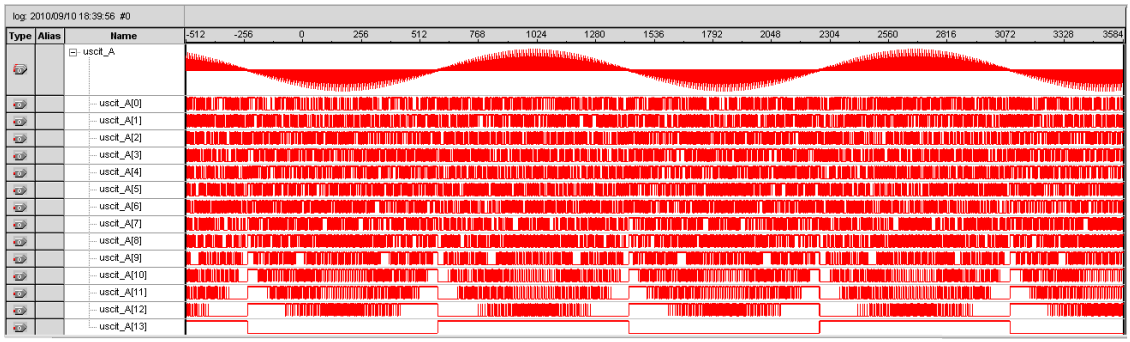

Fig. 4.12: Visualizzazione dell'andamento dei singoli bit dell'accumulatore

Capitolo 5

ANALISI TEORICA DEL SISTEMA DI ACQUISIZIONE DELLE TENSIONI

In questo capitolo si cercherà di analizzare, dal punto di vista teorico, le prestazioni del sistema di acquisizione delle tensioni, la cui realizzazione è stata affrontata nei capitoli precedenti. In particolare verrano esplicati i passaggi utilizzati ai fini di ottenere l'attenuazione totale del sistema, e la sensibilità in ingresso del sistema stesso.

5.1 Sistema di acquisizione a 20Mhz

5.1.1 Attenuazione

Si immagina di avere in ingresso al sistema la massima tensione positiva pari a +300V, lo stadio di ingresso la attenua (di un fattore 1:200) portandola a +1,5V; la scheda di interfaccia introduce un'ulteriore attenuazione, (pari ad un fattore 1:1,5) in maniera da ottenere in ingresso al converitore AD una tensione differenziale del valore di +1V (+0,5V su un canale differenziale e -0,5V sull'altro). Tale valore di tensione è il massimo positivo accetato in ingresso all'ADC, che pertanto lo converte in digitale, con il corrispondente massimo numero positivo esprimibile in Offset Binary a 14bit, che, convertito in decimale è dato dalla 5.1.

$$+2^{14-1} - 1 = +8191 \tag{5.1}$$

Tale numero è portato in ingresso ad uno dei due accumulatori del programma caricato nell'FPGA. Pertanto l'uscita dell'accumulatore corrisponderà al numero decimale dato dalla 5.2.

$$+8191 \cdot (20Mhz/10Khz) = +8191 \cdot 2000 = +16382000 \tag{5.2}$$

A questo punto il risultato dell'accumulazione viene troncato per ottenere i 14bit di uscita. Per passare da 25bit a 14bit occorre effettuare uno shift logico a destra di 11 posizioni; il che equivale a dividere il numero decimale ottenuto per $2^{11} = 2048$. Quindi si ottiene il risultato dato dalla 5.3.

$$\frac{+16382000}{2048} = +7999,02344 \tag{5.3}$$

Del quale si deve considerare il valore intero +7999, che, entrando in ingresso al convertitore DA, viene trasformato in un segnale analogico. Come visto alla fine della sottosezione 2.4.3 la sensibilità d'uscita del DAC è pari a 58,594μV. Pertanto, il valore di tensione d'uscita corrispondente al numero +7999, è dato dalla 5.4.

$$+7999 \cdot 58,594\mu V + V_{OFFSET} = 468,693mV + 480mV \tag{5.4}$$

Poiché per il calcolo dell'attenuazione totale (rispetto ai +300V di ingresso) non bisogna considerare l'Offset; l'attenuazione introdotta dall'intero sistema è data dalla 5.5.

$$\frac{300V}{468,693mV} = 640,078 \tag{5.5}$$

Si fa notare che nel calcolo dell'attenuazione totale del sistema, le conversioni da OB a C2 e viceversa presenti nel programma, risultino del tutto ininfluenti, e quindi non sono state prese in considerazione nei passaggi affrontati.

5.1.2 Sensibilità

Il numero N di campioni digitalizzati dal convertitore AD in ogni periodo PWM è dato dalla 5.6.

$$N = \frac{20Mhz}{10Khz} = 2000 \; campioni \; per \; periodo \; PWM \tag{5.6}$$

Come visto anche sopra, il numero T equivalente alla divisione decimale, corrispondente al troncamento dei dati in uscita dall'accumulatore, è dato dalla 5.7.

$$T = 2^{11} = 2048 \tag{5.7}$$

Come precedentemente accennato in sottosezione 3.4.1, poiché il numero T risulta strettamente maggiore del numero N, si avrà una diminuzione della sensibilità del sistema.

Verranno di seguito riportati i passaggi effettuati a ritroso, a partire dal dato digitale di uscita del programma, ai fini di calcolare la sensibilità del sistema stesso.

Minimo numero positivo in uscita dopo il troncamento da 25bit a 14bit e trasformazione da C2 a OB:

OB: 1 0 0 0 0 0 0 0 0 0 0 0 0 1

prima del troncamento:

OB: 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

prima della trasformazione da C2 a OB:

C2: 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

conversione decimale del numero:

DEC: 2048 $(= 2^{11})$

Questo è il numero salvato nel registro di uscita dell'accumulatore, poiché esso in un periodo effettua 2000 sommatorie, allora, il numero più piccolo in ingresso all'accumulatore affinché io veda in uscita un 1 (decimale), è dato dalla 5.8.

$$\frac{T}{N} = 2048/2000 = 1,024 \tag{5.8}$$

Ovviamente il converitore AD trasforma segnali analogici in valori discreti, quindi non può convertire un dato di ingresso con un valore di uscita pari a 1,024. Bensì, può produrre un 1, che se accumulato 2000 volte, realizzerebbe un zero in uscita. Oppure potrebbe produrre un 2, che, se accumulato 2000 volte realizzerebbe un'uscita positiva. Ciò potrebbe far pensare che il 2 sia il numero istantaneo minimo in ingresso che fa sì che il sistema produca un uscita positiva. Ragionando in questo modo si fa un errore di valutazione perché non si ragiona sul valore medio in ingresso. Di seguito si spiega meglio tale affermazione.

In realtà il minimo numero totale accumulato che fa scattare un uno in uscita è il 2048, che, come dato dalla 5.8, corrisponde al numero di ingresso 1,024. Ma questo numero rappresenta il valore medio, che se entrasse nell'accumulatore, per 2000 accumulazioni, produrrebbe il numero 2048, visibile come un uno in uscita dopo il troncamento. Quindi nell'esempio appena proposto il numero 2048 viene prodotto da un'accumulazione di 2000 campioni di valore 1,024 non rappresentabili dall'ADC. Ma tale risultato può essere raggiunto da una sommatoria di 2000 campioni interi (con valore medio 1,024) rappresentabili dall'ADC. Quindi il numero corretto da prendere in considerazione nei seguenti passaggi è proprio 1,024.

Poiché il numero massimo di uscita dall'ADC, convertito in decimale, è dato dalla 5.1. Ed essendo tale valore corrispondente ad una tensione di ingresso pari ad +1V. La risoluzione di ingresso dell'ADC è data dalla 5.9.

$$\frac{+1V}{+8191} = 122,085\mu V \tag{5.9}$$

Pertanto la sensibilità in ingresso all'FPGA è data dalla 5.10

$$122,085\mu V \cdot 1,024 = 125,015\mu V \tag{5.10}$$

Ciò significa che il programma caricato nell'FPGA è sensibile a tensioni medie in ingresso pari a $125,015\mu V$.

Infine ricordando che, prima di giungere al convertitore AD, il segnale subisce due attenuazioni pari a 1:1,5 e 1:200.

L'equazione 5.11 fornisce la sensibilità del sistema completo.

$$125,015\mu V \cdot 1,5 \cdot 200 = 37,505mV \tag{5.11}$$

Ricapitolando, affinché in uscita dall'accumulatore sia presente un 1; in ingresso al sistema deve esserci un segnale con valore medio sul periodo (100μs) maggiore o uguale a 37,505 mV.

5.2 Sistema di acquisizione a 1,28Mhz

Oltre al programma caratterizzato da una frequenza di campionamento del convertitore AD pari a 20Mhz, è stata successivamente realizzata un'altra versione con campionamento pari a 1,28Mhz. La frequenza di campionamento è stata scelta in maniera da ottenere l'uguaglianza $T = N$, in modo che il programma caricato nell'FPGA fornisse, in uscita, effettivamente il valore medio dei dati in ingresso all'FPGA e non un valore proporzionale. Di seguito viene approfondito tale passaggio.

Il numero N di campioni digitalizzati dal convertitore AD in ogni periodo PWM è dato dalla 5.12.

$$N = \frac{1,28Mhz}{10Khz} = 128\ campioni\ per\ periodo\ PWM \tag{5.12}$$

L'equazione 5.1 fornisce il massimo numero rappresentabile in uscita dall'ADC (convertito in decimale). Se per un intero periodo PWM tale valore entrasse in un accumulatore, il registro di uscita dell'accumulatore stesso dovrebbe contenere il numero decimale dato dalla 5.13.

$$+8191 \cdot 128 = 1048448 = max\ N°accumulabile \tag{5.13}$$

Tale numero risulta rappresentabile con 21bit, infatti si ottiene: $2^{21-1} - 1 = 1048575$ che risulta maggiore del numero ricavato dalla 5.13. Pertanto il registro di uscita di ciascun accumulatore deve contenere 21bit.

Poiché l'ingresso del convertitore DAC accetta segnali a 14bit, deve essere effettuata un'operazione di troncamento dei 7 bit meno significativi dell'uscita degli accumulatori. Con questo dimensionamento il numero T, equivalente alla divisione decimale, corrispondente al troncamento dei dati in uscita dall'accumulatore, è dato dalla 5.14.

$$T = 2^7 = 128 \tag{5.14}$$

Che, come ci si aspettava, corrisponde al numero N di campioni processati dal programma in ciascun periodo PWM (vedi 5.12). Vedremo come tale accortezza produca un leggero migliramento della sensibilità del sistema.

5.2.1 Attenuazione

Come per il calcolo dell'attenuazione del sistema a 20Mhz si immagina di avere in ingresso al sistema la massima tensione positiva pari a +300V, nella 5.15 si schematizzano i passaggi, già ampiamente spiegati:

$$(+300V) \rightarrow (+1,5V) \rightarrow (+1V) \rightarrow (+8191) \rightarrow (+8191 \cdot 128) = +1048448 \tag{5.15}$$

A questo punto il risultato dell'accumulazione viene troncato per ottenere i 14bit di uscita. Per passare da 21bit a 14bit occorre effettuare uno shift logico a destra di 7 posizioni; il che equivale a dividere il numero decimale ottenuto per $2^7 = 128$. Quindi si ottiene il risultato dato dalla 5.16.

$$\frac{+1048448}{128} = +8191 \tag{5.16}$$

Che ovviamente risulta uguale al numero convertito dall'ADC in ingresso al programma, poiché il valore medio di un numero costante in ingresso è uguale al numero stesso. Il numero +8191 entrando in ingresso al convertitore DA, viene trasformato in un segnale analogico. Come riportato prima, la sensibilità d'uscita del DAC è pari a 58,594μV. Pertanto, il valore di tensione d'uscita corrispondente al numero +8191, è dato dalla 5.17.

$$8191 \cdot 58,594\mu V + V_{OFFSET} = 479,943mV + 480mV \tag{5.17}$$

Poiché per il calcolo dell'attenuazione totale (rispetto ai +300V di ingresso) non bisogna considerare l'Offset; l'attenuazione introdotta dall'intero sistema è data dalla 5.18.

$$\frac{300V}{479,943mV} = 625,074 \tag{5.18}$$

L'attenuazione risulta inferiore rispetto a quella introdotta dal sistema con campionamento a 20Mhz. Il che risulta corretto, in quanto, mentre nel sistema con campionamento a 1,28Mhz il dato d'uscita decimale è +8191, nel sistema con campionamento a 20Mhz, il dato decimale d'uscita è +7999. Tale risultato viene convertito in un valore di tensione inferiore rispetto alla conversione analogica del numero +8191. Pertanto il sistema con campionamento a 1,28Mhz è caratterizzato da un'attenuazione inferiore rispetto a quello con campionamento a 20Mhz.

5.2.2 Sensibilità

Il numero N di campioni digitalizzati dal convertitore AD in ogni periodo PWM è dato dalla 5.12.

Il numero T, equivalente alla divisione decimale, corrispondente al troncamento dei dati in uscita dall'accumulatore, è dato dalla 5.14. Pertanto, come già fatto notare, N e T sono entrambi uguali a 128.

Come effettuato per la versione del programma a 20Mhz, di seguito vengono riportati i passaggi effettuati a ritroso, a partire dal dato digitale di uscita del programma, ai fini di calcolare la sensibilità del sistema stesso.

Minimo numero positivo in uscita dopo il troncamento da 21bit a 14bit e trasformazione da C2 a OB:

OB: 1 0 0 0 0 0 0 0 0 0 0 0 0 1

prima del troncamento:

OB: 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

prima della trasformazione da C2 a OB:

C2: 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

conversione decimale del numero:

DEC: 128 ($= 2^7$)

Questo è il numero salvato nel registro di uscita dell'accumulatore, poiché esso in un periodo effettua 128 sommatorie, allora, il numero più piccolo in ingresso all'accumulatore affinché io veda in uscita un 1 (decimale), è dato dalla 5.19.

$$\frac{T}{N} = \frac{128}{128} = 1 \tag{5.19}$$

La risoluzione di ingresso dell'ADC è data dalla 5.9. Per calcolare la sensibilità in ingresso all'FPGA dovrei moltiplicare pertanto il valore 122,085 μV per il rapporto T/N. Essendo tale rapporto uguale a uno, la risoluzione dell'ADC coincide con la sensibilità di ingresso dell'FPGA. Quindi il programma caricato nell'FPGA è sensibile a tensioni medie di ingresso pari a 122,085 μV.

Infine ricordando che, prima di giungere al convertitore AD, il segnale subisce due attenuazioni pari a 1:1,5 e 1:200. L'equazione 5.20 fornisce la sensibilità del sistema completo.

$$125,085\mu V \cdot 1,5 \cdot 200 = 37,526mV \tag{5.20}$$

Ricapitolando, affinché in uscita dall'accumulatore sia presente un 1; in ingresso al sistema deve esserci un segnale con valore medio sul periodo (100μs) maggiore o uguale a 37,526 mV.

Il miglioramento ottenuto (rispetto al sistema con campionamento a 20Mhz) non è molto significativo a livello di performance. Ma utile per capire che la sensibilità del sistema di ingresso risulta inversamente proporzionale al rapporto T/N. Se tale rapporto risulta essere uguale a uno, la sensibilità del sistema dipende soltanto dalla risoluzione di

ingresso del convertitore AD (e ovviamente dagli stadi di riduzione). All'aumentare di tale rapporto la risoluzione progressivamente cala, a causa della penalizzante operazione di troncamento.

Capitolo 6

RISULTATI SPERIMENTALI

In questo capitolo verrano riportati i risultati ottenuti dalle misurazioni sperimentali effettuate sul sistema di misura delle tensioni dell'inverter. Verranno illustrate le differenze calcolate tra le misurazioni reali e i valori teorici. Verranno inoltre confrontati il sistema con campionamento a 20Mhz e il sistema con campionamento a 1,28Mhz.

6.1 Misure in continua

Per prima cosa sono state misurate le prestazioni del sistema, ponendo in ingresso una tensione continua all'interno del range $\pm 300V$. In particolare sono stati misurati i valori di uscita del sistema, fornendo delle tensioni di ingresso all'interno del range (riportato poco fa) con intervalli di 10V.

Dalle misurazioni così ottenute, è stato ricavato il valore medio di attenuazione totale, e successivamente la sua deviazione standard. Poi si è passati ad analizzare la linearità del sistema. Con Excel è stato ricavato il coefficiente angolare del sistema linearizzato, come riportato nella Fig. 6.1 (sistema con campionamento a 20Mhz) e nella Fig. 6.2 (sistema con campionamento a 1,28Mhz). Grazie al quale è stata calcolata l'entità dello scostamento del sistema reale da quello linearizzato. In particolare sono stati ricavati l'errore massimo e l'errore medio percentuali.

6.1.1 Sistema con campionamento a 20Mhz

Il valore medio dell'attenuazione totale e la deviazione standard, del sistema con campionamento a 20Mhz, sono riportati rispettivamente nella 6.1, e nella 6.2.

$$ATTENUAZIONE\ MEDIA = 635,205 \tag{6.1}$$

Tale valore di attenuazione si discosta di 0,76% dai calcoli teorici visti in sottosezione 5.1.1.

$$DEVIAZIONE\ STANDARD = 2,164V \tag{6.2}$$

Il coefficiente angolare del sistema linearizzato, ottenuto dal grafico in Fig. 6.1, risulta uguale a 0,0015734693 (precisione fino alla decima cifra decimale).

Dal quale (attraverso una semplice moltiplicazione) sono stati ricavati tutti i valori di uscita linearizzati (a partire dalle tensioni di ingresso continue misurate).

Infine dalla 6.3 sono stati ricavati tutti gli errori percentuali tra le uscite misurate e le uscite linearizzate.

$$E_{\%} = \frac{OUT_{misurata} - OUT_{linearizzata}}{OUT_{misurata}} \cdot 100 \tag{6.3}$$

In modo da ottenere l'errore massimo e l'errore medio percentuali riportati rispettivamente nelle 6.4 e 6.5.

$$E_{max\ \%} = 1,109\% \tag{6.4}$$

$$E_{mean\ \%} = 0,053\% \tag{6.5}$$

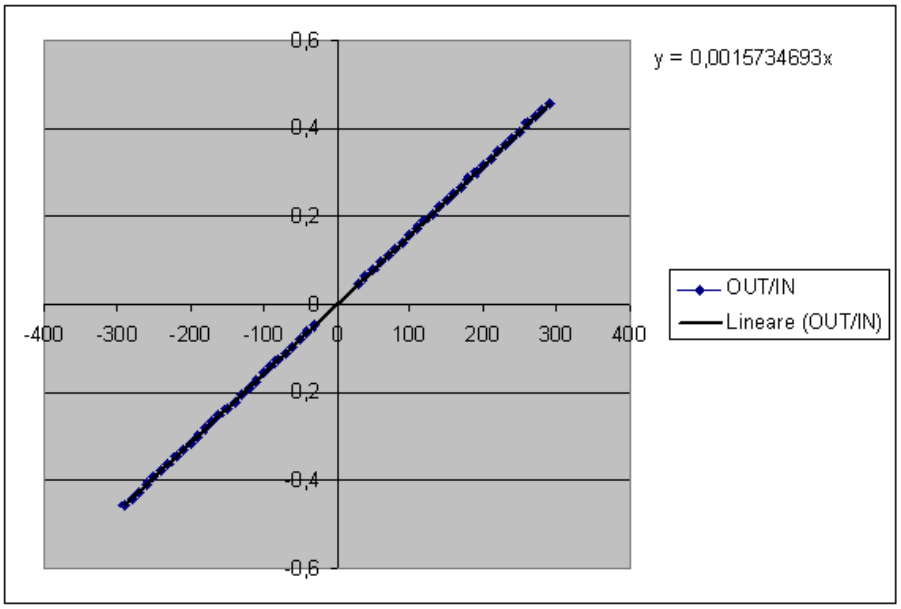

Fig. 6.1: Linearizzazione del sistema con campionamento a 20Mhz

6.1.2 Sistema con campionamento a 1,28Mhz

Il valore medio dell'attenuazione totale e la deviazione standard, del sistema con campionamento a 1,28Mhz, sono riportati rispettivamente nella 6.6, e nella 6.7.

$$ATTENUAZIONE\ MEDIA = 616,133 \tag{6.6}$$

Tale valore di attenuazione si discosta di 1,45% dai calcoli teorici visti in sottosezione 5.1.1.

$$DEVIAZIONE\ STANDARD = 0,471V \tag{6.7}$$

Il coefficiente angolare del sistema linearizzato, ottenuto dal grafico in Fig. 6.2, risulta uguale a 0,0016224730 (precisione fino alla decima cifra decimale).

Grazie al quale, come in sottosezione 6.1.1, sono stati ricavati l'errore massimo e l'errore medio percentuali (tra i valori misurati e i valori linearizzati) riportati rispettivamente nelle 6.8 e 6.9.

$$E_{max\ \%} = 1,170\% \tag{6.8}$$

$$E_{mean\ \%} = 0,034\% \tag{6.9}$$

6.1.3 Confronto tra i due sistemi

Dai risultati ottenuti si può notare come le prestazioni dei due sistemi siano confrontabili. In particolare gli errori massimi e medi rispetto ai sistemi linearizzati risultano molto vicini tra loro.

Il sistema con campionamento a 1,28Mhz ha una deviazione standard pari a 0,471V che risulta migliore di quella pari a 2,164V, calcolata per il sistema con campionamento a 20Mhz.

Si ricorda (come visto nel capitolo 5) che il sistema con campionamento a 1,28Mhz fornisce il valore medio digitale dei dati (anch'essi digitali) in ingresso al programma. Invece il sistema con campionamento a 20Mhz fornisce un valore proporzionale al valore medio.

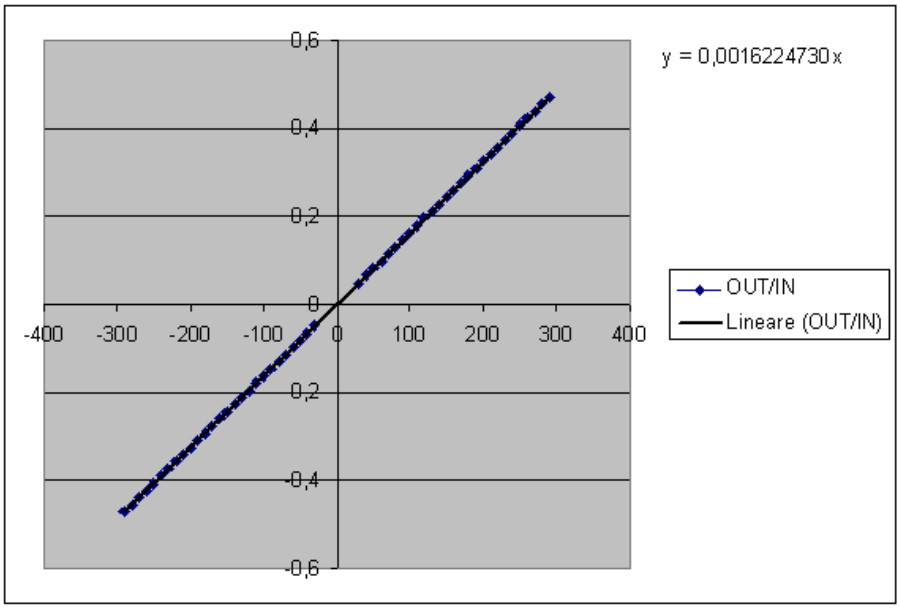

Fig. 6.2: Linearizzazione del sistema con campionamento a 1,28Mhz

Infine il sistema con campionamento a 1,28Mhz è dotato della migliore sensibilità possibile, dipendente soltanto dalla risoluzione del convertitore AD e dagli stadi di attenuazione di ingresso.

Per tanto si può concludere che il sistema con campionamento a 1,28Mhz risulta maggiormente ottimizzato rispetto a quello con campionamento a 20Mhz.

6.2 Misurazioni sperimentali delle tensioni in inverter trifase

Nelle figure Fig. 6.3 e Fig. 6.4 si riportano i risultati sperimentali ottenuti, relativi ad una sola tensione concatenata del motore. Le figure sono rispettivamente ricavate utilizzando il sistema con campionamento a 20Mhz e il sistema con campionamento a 1,28Mhz.

In particolare un motore sincrono a magneti permanenti, è stato alimentato con una tensione di fase di ampiezza 70V, e una frequenza di 50Hz (tramite un controllo V/Hz). La frequenza di modulazione PWM utilizzata è pari a 10Khz. Il segnale di colore blu rappresenta la tensione di alimentazione concatenata fornita al motore, mentre quello di colore nero rappresenta la tensione media ottenuta dall'elaborazione del sistema di acquisizione, infine quello di colore rosso rappresenta la tensione media ottenuta con Matlab.

In entrambe le figure si può osservare che il risultato dell'elaborazione del sistema di acquisizione delle tensioni, risulta molto aderente al calcolo ideale delle tensioni medie ottenuto con matlab. Tanto che risulta utile per entrambe le figure, visualizzare degli zoom per apprezzare maggioramente i due risultati, che nella visione di insieme sembrano praticamente sovrapposti.

A tale scopo le figure Fig. 6.5 e Fig. 6.6 riportano gli ingrandimenti relativi rispettivamente al sistema con campionamento a 20Mhz e al sistema con campionamento a 1,28Mhz. Dai quali si può notare come entrambe le soluzioni seguano bene il calcolo teorico implementato con Matlab.

Negli ingrandimenti è stato catturato un punto caratterizzato da una non linearità causata dall'introduzione dei tempi morti negli interruttori dell'inverter, come visto

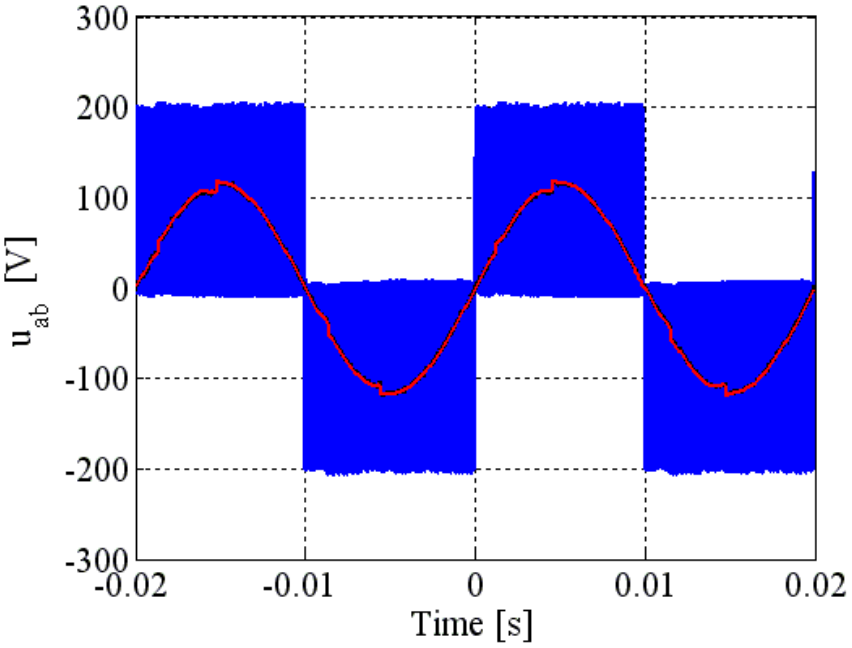

Fig. 6.3: Tensione Media Concatenata (Campionamento a 20Mhz)

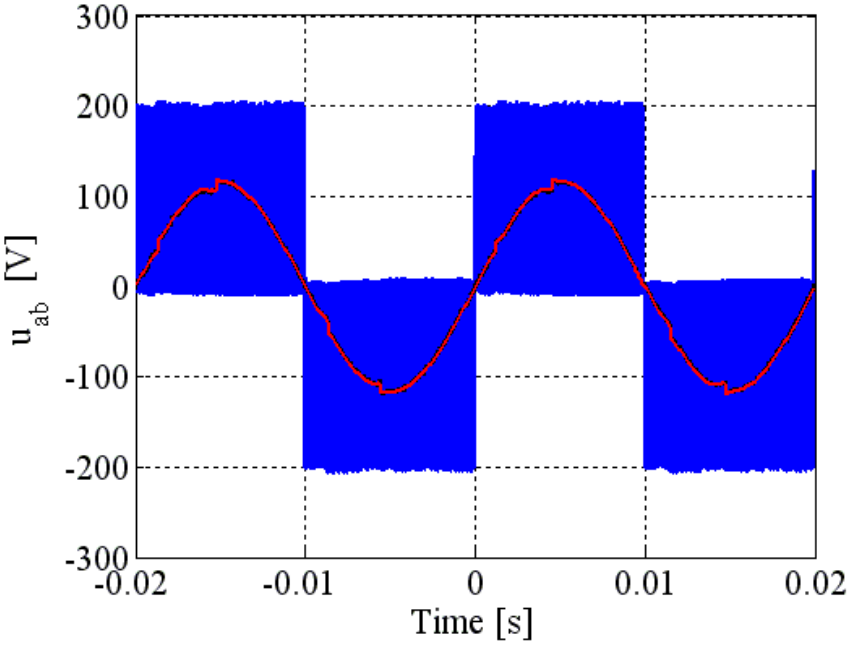

Fig. 6.4: Tensione Media Concatenata (Campionamento a 1,28Mhz)

all'inizio della tesi nel capitolo 1. Nelle 6.10 e 6.11 vengono riportati gli errori medi percentuali tra il calcolo effettuato con matlab e le misurazioni sperimentali.

$$ERRORE_{\%\ (20Mhz)} = 0,18\% \tag{6.10}$$

$$ERRORE_{\%\ (1,28Mhz)} = 0,71\% \tag{6.11}$$

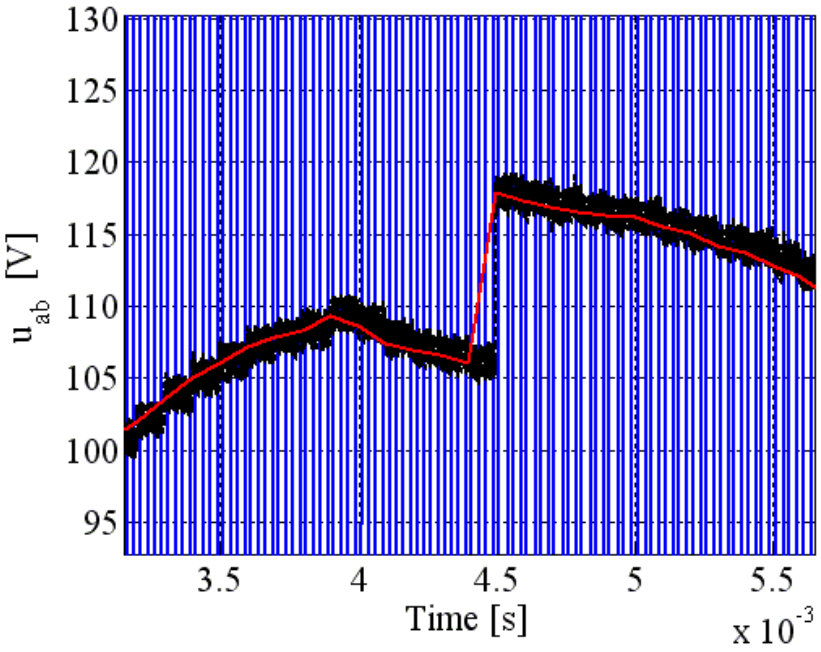

Fig. 6.5: Tensione Media Concatenata (Campionamento a 20Mhz) (Ingrandimento)

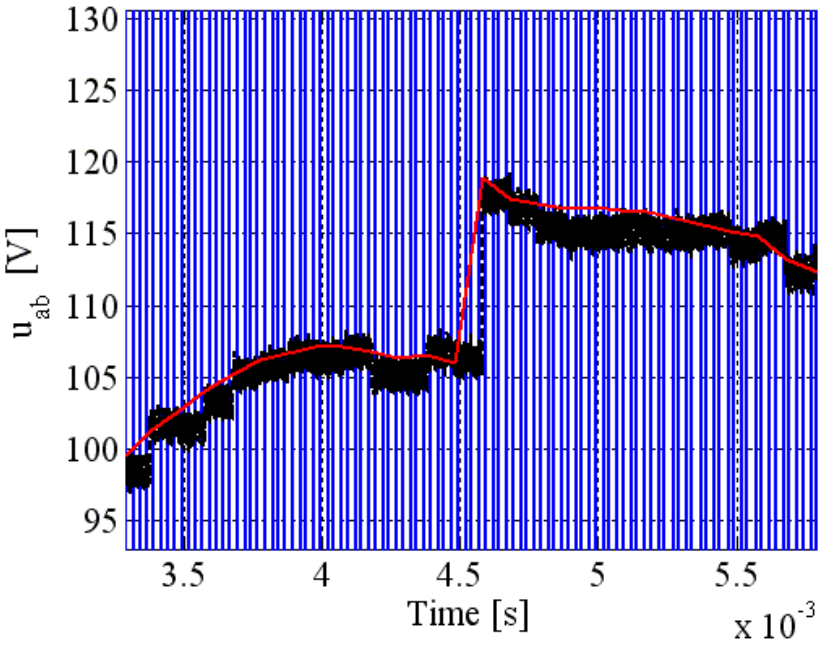

Fig. 6.6: Tensione Media Concatenata (Campionamento a 1,28Mhz) (Ingrandimento)

Di seguito vengono infine riportate le precise condizioni di misura delle prove sperimentali insieme alle caratteristiche degli strumenti utilizzati.

programma dSPACE: \ codiceDSPACE \ FOC-IM
funzionamento IM in V/Hz
Bus in continua: 200 V
frequenza 50 hz [dSPACE]

fswitching= 10 kHz
ampiezza tensione di fase 70 V [dSPACE]
Oscilloscopio Tektronix DPO4034
Time windows: 0,04s [4 ms/div]
Acquisition lenght: 1Ms (su tutto lo schermo, time window)
Mode: high resolution

canale tensione in ingresso:
sonda differenziale: tektronix P5205 (attenuazione x500)
asse y : 50 V/div

canale d'uscita:
asse y: 500 mV/div

Canale di sincronismo:
asse y: 500 mV/div

CONCLUSIONI E SVILUPPI FUTURI

L'obiettivo di realizzare un sistema di acquisizione delle tensioni fondamentali di un inverter trifase si può dire raggiunto. Da qui in poi nascono numerose strade da poter percorrere, sia per quanto riguarda il miglioramento del progetto stesso, sia per quanto riguarda l'applicazione dei risultati ottenuti.

Il primo è il miglioramento del software. Il programma infatti, è perfettamente funzionante, ma è tarato su una particolare condizione di lavoro. Vale a dire che è dimensionato per una particolare frequenza di modulazione PWM. Se la frequenza di PWM cambiasse, bisognerebbe verificare se il software è in grado di supportare in maniera corretta le nuove impostazioni. In particolare con frequenze di PWM più basse, gli accumulatori lavorerebbero per un periodo più lungo andando incontro a possibili errori di overflow. Mentre se la frequenza della PWM aumentasse gli accumulatori sarebbero attivi per un periodo più breve. Il che porterebbe a valori finali di accumulazione più piccoli. Quindi, in tal caso, il troncamento finale risulterebbe più pesante. Il che potrebbe portare a una diminuzione della sensibilità del sistema.

Pertanto si è costretti ad adattare di volta in volta il sistema a lavorare alle varie frequenze PWM desiderate. In questo modo si ha un cambiamento dei parametri quali sensibilità, attenuazione, linearità, etc. del sistema stesso.

Una soluzione teoricamente realizzabile consisterebbe nel fornire una frequenza di campionamento non fissata in maniera rigida, ma variabile. In particolare multipla della frequenza della PWM.

Per esempio si è osservato che il sistema con campionamento a 1,28Mhz fornisce delle ottime prestazioni se applicato ad una frequenza della PWM pari a 10Khz. In particolare in queste condizioni si è fatto notare che, essendo il totale delle accumulazioni pari a 128, ed essendo il troncamento finale dei registri degli accumulatori equivalente ad una divisione per 128, il risultato digitale di uscita dal programma corrisponde al valore medio reale dei dati digitali in ingresso al programma stesso. Quindi, per ottenere gli stessi risultati a fronte di frequenze della PWM diverse, bisognerebbe, in linea teorica, far si che il campionamento dei dati dell'ADC (e ovviamente il conteggio negli accumulatori) fosse caratterizzato da una frequenza pari a 128 volte la frequenza della PWM. In questo modo si otterrebbe un sistema dalle caratteristiche quali sensibilità, attenuazione, linearità, etc. fisse, (e non più variabili) funzionante qualunque sia la frequenza della modulazione PWM.

Ovviamente questo è vero finché la frequenza della PWM non risulti molto bassa, rispetto al dimensionamento scelto per il sistema. Infatti, se questo accadesse, il periodo di accumulazione diventerebbe molto lungo. In questa condizione occorre verificare se i 128 punti rimangono sufficienti ad ottenere il valore medio nel periodo PWM, con una certa accuratezza.

In linea teorica tale sviluppo è fattibile, anche se con gli strumenti (del `Quartus II`) fin qui utilizzati, non risulta di immediata implementazione.

Il secondo aspetto da affrontare sarà l'utilizzo delle tensioni misurate (digitali) all'interno di algoritmi sensorless per la stima di coppia. In Fig. 6.7 è riportato uno schema a blocchi rappresentante l'implementazione di un algoritmo per la sitma della coppia in un motore sincrono a magneti permanenti ottenuto da [16]. In figura si può notare come i blocchi, evidenziati in rosso, denominati con la lettera `M` siano predisposti al calcolo delle tensioni fondamentali concatenate. Quindi tali blocchi rappresentano proprio il sistema sviluppato in questo progetto.

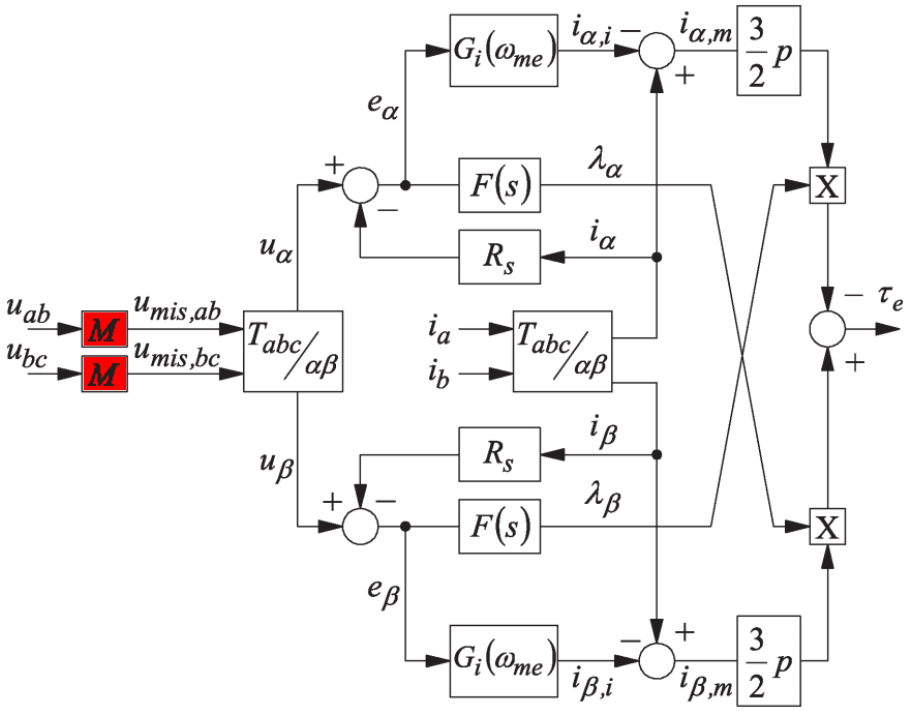

Fig. 6.7: Algoritmo sensorless per la stima della coppia

Infine verrà affrontata la creazione di un'unica scheda, contenente l'hardware di acquisizione e trasformazione analogica/digitale delle tensioni dell'inverter. Il sistema di acquisizione, come ogni prodotto nel mercato, dovrà essere realizzato cercando il miglior compromesso tra prestazioni e costo economico. A tale fine si fanno alcune considerazioni.

Inizialmente si è pensato di lavorare con un campionamento a 20Mhz per verificare l'effettiva realizzabilità del sistema. In questo modo è stata sfruttata l'elevata velocità di campionamento dei convertitori AD presenti nella `Terasic_THDB_ADA`. Ciò, nell'ottica di realizzare il sistema di acquisizione in un'unica scheda, comporterebbe l'utilizzo di due ADC dal costo compreso tra i 50 e i 100€. In una fase successiva si è passati a lavorare ad una frequenza molto più bassa pari a 1,28Mhz. Con risultati sicuramente paragonabili a quelli ottenuti con un campionamento a frequenza più elevata, se non migliori. Il che comporterebbe l'utilizzo di un ADC dal costo inferiore ai 20€, come per esempio il modello ADS7863, il cui datasheet è reperibile in [17]. La scelta di lavorare a frequenze di campionamento più basse, permette quindi di ridurre i costi mantenendo le prestazioni del sistema. Infine, come visto in sottosezione 3.5.12, il programma sfrutta una minima parte delle risorse hardware della scheda FPGA, il che permetterebbe di utilizzare una FPGA meno evoluta e quindi, più economica.

RINGRAZIAMENTI

Ringrazio tutte le persone che mi sono state vicine in questi anni di università e mi hanno aiutato a raggiungere il traguardo finale. In particolare il Prof. Mauro Zigliotto, l'Ing. Luca Peretti e tutti i colleghi e amici del laboratorio di Vicenza, che mi hanno fornito un aiuto prezioso per la realizzazione di questa tesi. Un ringraziamento speciale ai miei cari, agli amici e alla fidanzata, senza i quali non sarei riuscito a superare i momenti di difficoltà.

Bibliografia

[1] L. Malesani, "Dispensa del corso di Elettronica per l'Energia 2008-2009," Padova.

[2] M. Zigliotto, "Dispensa del corso di Azionamenti Elettrici 2009-2010," Vicenza.

[3] A. Devices, "Datasheet ad8138," http://www.analog.com/static/imported-files/data_sheets/AD8138.pdf.

[4] "cad freeware kicad," http://www.lis.inpg.fr/realise_au_lis/kicad/.

[5] N. Semiconductor, "Datasheet cd4538b dual precision monostabile," http://pdf1.alldatasheet.com/datasheet-pdf/view/80443/NSC/CD4538.html.

[6] A. Devices, "Datasheet ad9248," http://www.analog.com/static/imported-files/data_sheets/AD9248.pdf.

[7] Terasic, "Schematico thdb_ada," presente nel CD fornito con la scheda.

[8] A. Devices, "Datasheet ad9767," http://www.analog.com/static/imported-files/data_sheets/AD9763_9765_9767.pdf.

[9] Altera, "Reference manual cyclone_3_fpga_starter_board," http://www.altera.com/literature/manual/rm_ciii_starter_board.pdf.

[10] ——, "Quartus2 download page," https://www.altera.com/download/software/quartus-ii-we.

[11] ——, "Quartus2 manual," http://www.altera.com/literature/manual/intro_to_quartus2.pdf.

[12] "Manuale vhdl," http://web.ticino.com/pagna/Pagine/Documentazioni.

[13] Altera, "Schematico cyclone_3_fpga_starter_board," presente nel CD fornito con la scheda.

[14] ——, "Istruzioni installazione usb blaster," http://www.altera.com/download/drivers/dri-index.html.

[15] ——, "Driver usb blaster download," http://www.altera.com/support/software/drivers.

[16] B. Silverio, L. Peretti, M. Zigliotto, and E. Bertotto, "Commissioning of Electromechanical Conversion Models for High Dynamic PMSM Drives," in *IEEE Transaction on Industrial Electronics*, vol. 57, no. 3, March 2010.

[17] T. Instrument, "Datasheet ads7863," http://it.farnell.com/texas-instruments/ads7863irget/ic-adc-12bit-2msps-qfn-24/dp/1622642.