UNIVERSITÀ DI PISA
SCUOLA DI INGERNERIA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

Laurea Magistrale

Implementazione su FPGA di un algoritmo per la stima online dello stato di batterie al litio

Candidato: Rocco Morello

Relatore: Prof. Roberto Saletti ….....

Relatore: Prof. Federico Baronti ….....

Relatore: Ing. Roberto Di Rienzo ….....

Indice

Introduzione....3

1 Il Battery Management System (BMS)....5
1.1 Le batterie al litio....5
1.2 Le funzioni del Battery Management System....8
1.3 Tipologie e architetture del BMS....10

2 Stima dello stato della batteria....14
2.1 Lo stato di carica (SOC)....14
2.2 Metodi di stima del SOC....16
2.2.1 Coulomb Counting....16
2.2.2 Discharge test....18
2.2.3 Open circuit voltage....18
2.2.4 Neural Network....19
2.2.5 Model based....20
2.2.6 Mixed Algorithm....24
2.3 Confronto tra i vari metodi....24
2.4 L'identificazione online dei parametri nei modelli....25
2.5 Modelli per l'identificazione....26
2.6 Metodo dei minimi quadrati....28
2.7 Algoritmi per la risoluzione dei problemi di minimi quadrati....31
2.7.1 Fattorizzazione QR....31
2.7.2 Fattorizzazione Cholesky....33
2.7.3 Confronto tra gli algoritmi trattati....34
2.8 Stato dell'arte sulla realizzazione degli algoritmi di stima....35

3 Algoritmo di stima proposto....37
3.1 Descrizione generale....37
3.2 Algoritmo di stima del SOC....38
3.2.1 Modello della batteria....40
3.3 Analisi della stabilità e della sensibilità agli errori....41
3.3.1 Analisi sulla stabilità....42
3.3.2 Analisi sugli errori di misura....44
3.3.3 Comportamento in presenza di un'inizializzazione errata del SOC....46
3.4 Algoritmo di stima online dei parametri....47
3.4.1 Caratterizzazione del sistema....48
3.4.2 Descrizione e funzionamento dell'algoritmo....53
3.4.3 Fattorizzazione QR con Givens rotation....56

4 Implementazione dell'algoritmo....60
4.1 Flusso di progetto....60

4.2 Descrizione generale del sistema....63
4.3 Stima del SOC....65
4.4 Identificazione online dei parametri....71
4.4.1 Window....74
4.4.2 Vector_Mem....76
4.4.2 Array_Compiler....77
4.4.2 QRD....78
4.4.2 Final_Calc....80

5 Validazione dell'implementazione dell'algoritmo....86
5.1 Descrizione generale del test....86
5.2 Scheda di sviluppo SoCKit....87
5.2.1 Occupazione delle risorse dell'FPGA....90
5.3 Descrizione del sistema di verifica....91
5.4 Caratterizzazione della batteria....93
5.4.1 Caratteristica OCV-SOC....94
5.4.2 Stima offline dei parametri....96
5.5 Urban Dynamometer Driving Schedule (UDDS)....99
5.6 Risultati del test....101
5.6.1 Modulo di stima del SOC....102
5.6.2 Modulo di identificazione online dei parametri....105
5.7 Prestazioni....107

Conclusioni....109

Bibliografia....111

Introduzione

Le batterie al litio stanno avendo una grande diffusione, grazie alla loro elevata densità di potenza e di energia, che le rendono particolarmente adatte anche all'utilizzo in veicoli elettrici o ibridi. La ricerca continua ancora ad interessarsi ad alcuni elementi delle batterie al litio, come i materiali che costituiscono le celle e i circuiti e gli algoritmi che si occupano del monitoraggio e controllo delle batterie. Infatti un utilizzo efficiente e sicuro di queste batterie si ha solo se esiste un BMS (Battery Management System). Esso si occupa, tramite opportuni algoritmi, di evitare situazioni in cui si hanno valori di tensione e temperatura dannosi per la batteria, di tenere traccia dello stato di carica (SOC, State Of Charge) e dello stato di salute (SOH, State Of Health) della batteria e di effettuare operazioni di manutenzione come il bilanciamento delle celle. Tutte le operazioni svolte dal BMS richiedono una stima il più accurata possibile del SOC e del SOH. Questi parametri non sono direttamente misurabili e si necessita quindi di un modello che descriva il comportamento sia statico che dinamico della batteria. La precisione di stima di SOC e SOH dipende dall'accuratezza del modello. Un tipico modello è costituito da resistenze, condensatori e generatori controllati che hanno parametri che dipendono dal tempo e dall'utilizzo, in particolare dal SOC, dal tasso di carica/scarica e dalla temperatura. Si necessita, quindi, di un sistema d'identificazione online di questi parametri.

Lo scopo di questa tesi è quello di realizzare su FPGA un sistema di stima dello stato di carica con identificazione online dei parametri per batterie al litio. La scelta di utilizzare

un FPGA deriva dal fatto che essi sono sempre più utilizzati per la realizzazione di DSP (Digital Signal Processor). In particolare, quando è possibile ottenere una parallelizzazione del flusso dei dati, gli FPGA consentono di ottenere un'elevata flessibilità e prestazioni. Si procederà quindi con una panoramica degli algoritmi scelti per realizzare questo sistema. Successivamente, verrà presentata la realizzazione dell'algoritmo e la sua implementazione su FPGA utilizzando un flusso di progetto innovativo. Infine, verranno presentati i risultati dei test svolti al fine di validare l'implementazione hardware dell'algoritmo.

1 Il Battery Management System (BMS)

Questo capitolo descrive le tipologie di batterie al litio disponibili in commercio e offre una descrizione generale del funzionamento e della struttura di un BMS.

1.1 Le batterie al litio

In commercio esistono numerosi tipi di batterie, diversi per materiali e tecnologia utilizzata. Tra queste troviamo le batterie al nichel-cloruro di sodio, al nichel-metallo idruro (Ni-MH) e agli ioni di litio (Li-ion). Le batterie al litio sono quelle che hanno avuto una più ampia diffusione negli ultimi anni grazie ai numerosi vantaggi rispetto alle altre tecnologie. Esse infatti possono essere costruite in varie forme e dimensioni, hanno un peso minore (gli ioni di litio hanno una densità di carica più elevata), non soffrono dell'effetto memoria, hanno un tasso di auto-scarica più basso, subiscono un deterioramento relativamente minore e hanno una tensione nominale più elevata [1]. Questi vantaggi rendono gli accumulatori al litio molto adatti all'utilizzo nei dispositivi portatili più comuni (come laptop, smartphone, tablet, videocamere, ecc.). Le batterie al litio vengono inoltre largamente utilizzate sui mezzi di trasporto elettrici e ibridi. Questa tipologia di accumulatori presenta però anche alcuni svantaggi come la sensibilità alla temperatura e il deterioramento che può avvenire in sotto-scarica e in sovraccarica. In particolare, in condizioni di sovraccarica, e in altre situazioni di stress elettrici, meccanici e termici, possono avvenire delle fughe di gas infiammabile del solvente presente nell'elettrolita, creando una situazione pericolosa sia per la cella che per l'utente.

Invece, se si viene a creare una situazione di sotto-scarica, solitamente quando la tensione della cella scende sotto 2 V, l'elettrodo negativo comincia a dissolversi nell'elettrolita, causando un deterioramento della cella. Per questi motivi si necessita di un circuito di controllo che monitori e renda sicuro il funzionamento della batteria.

Le celle agli ioni di litio più diffuse sono costituite da un anodo di grafite e da un catodo formato da un ossido litiato di un metallo di transizione, come l'ossido di cobalto o il fosfato di ferro. Questi elettrodi hanno una struttura intercalante all'interno della quale penetrano gli ioni di litio, che vanno ad occupare una posizione interstiziale. Tra gli elettrodi si interpone poi un separatore microporoso che evita il cortocircuito, ma che permette il passaggio degli ioni. Il flusso di ioni si dirige dall'anodo al catodo in condizioni di scarica (la direzione si inverte se si passa alla condizione di carica). Gli ioni si muovono in un elettrolita liquido, composto da solventi organici e sali di litio. Gli elettroni vengono raccolti da connettori metallici, di rame per l'anodo e di alluminio per il catodo. Tutto il sistema viene posto in un contenitore metallico rigido e robusto, a forma di cilindro o prisma, utile anche a limitare i danni che potrebbero verificarsi nelle condizioni pericolose prima descritte. La tensione nominale di una batteria litio-ione è pari a circa 3.6 V. I materiali più comuni per la realizzazione del catodo sono il litio-ossido di cobalto ($LiCoO_2$, LCO), il manganato di litio ($LiMn_2O_4$, LMO), il fosfato di ferro ($LiFePO_4$, LFP) e l'ossido di Nickel-Manganese-Cobalto ($LiNi_{1-y-z}Mn_yCO_zO_2$, NMC) [1]. Nella seguente tabella sono riassunte le caratteristiche principali delle varie tipologie sopra elencate.

Specifiche	LCO	LMO	LFP	NMC
Tensione nominale (V)	3.9	3.7	3.2-3.3	3.6-3.7
Tensione limite (V)	4.2	4.2	3.6	4.2
Cicli di vita	500	500-1000	1000-2000	1000-2000
Temperatura operativa	Media	Media	Buona	Buona
Energia specifica (Wh/kg)	155	100-120	160	200
Potenza specifica (C)	1	10-40	35	10
Fuga termica (°C)	150	250	270	210
Sicurezza	Scarsa	Media	Molto buona	Buona
Costo	Alto	Basso	Medio	Medio

Tabella 1: Caratteristiche delle batterie agli ioni di litio più diffuse [1].

Le batterie LCO hanno un costo relativamente alto a causa della scarsa disponibilità in natura di cobalto, al contrario delle LMO che sono le più economiche, grazie all'abbondante disponibilità di manganese. Queste ultime però hanno la capacità più bassa tra tutte. Nelle applicazioni per veicoli elettrici quindi si preferiscono le batterie LFP e NMC, entrambe molto utilizzate. La differenza tra le due tipologie è che le NMC hanno una capacità più elevata e una minore sicurezza.

La ricerca nel campo delle batterie al litio è molto attiva e indirizza i propri sforzi nello sviluppo di nuovi materiali, nel miglioramento dell'affidabilità, della capacità e delle prestazioni.

Se come elettrolita si utilizza un polimero solido, come il poliacrilinitrile, invece di un solvente organico liquido si parla di batteria al litio-polimero (Li-poly). Il polimero deve essere permeabile agli ioni e in questo caso funge anche da separatore tra gli elettrodi. La cella è composta da fogli flessibili laminati, disposti uno sull'altro (laminato polimerico). Questo tipo di tecnologia costruttiva contribuisce ad una maggiore densità energetica. Inoltre, offre una maggiore sicurezza visto che gli elettrodi non sono più in

contatto con un elettrolita composto da una soluzione liquida acquosa. Per questi motivi non è più necessario un contenitore rigido ed è quindi possibile ottenere celle di forma diversa e più leggere, con un processo di costruzione più semplice.

La maggiore sicurezza delle batterie al litio-polimero permette anche una semplificazione dei circuiti di controllo e quindi una riduzione del costo degli stessi.

1.2 Le funzioni del Battery Management System

Come già detto le batterie al litio necessitano di un circuito di controllo, generalmente chiamato Battery Management System (BMS). Le funzionalità di un BMS variano in base alle applicazioni e alla complessità del sistema in cui viene impiegato. Per sistemi semplici, specialmente con un basso numero di celle collegate in serie, si potrebbe utilizzare un BMS con funzioni essenziali, ottenendo una riduzione dei costi.

Le celle al litio sono ampiamente utilizzate nei veicoli elettrici (EV) e ibridi (HEV), grazie alla loro elevata densità energetica e di potenza. Per raggiungere le elevate tensioni (centinaia di volt) richieste in questo campo, e in generale nelle applicazioni high-power, è necessario collegare più celle in serie, creando così il pacco batteria.

Una soluzione di questo tipo porta con se numerose complicazioni. La batteria deve lavorare nella zona di funzionamento sicuro, deve quindi rispettare i range operativi di tensione e temperatura, per evitare situazioni pericolose o il degrado delle prestazioni.

Una delle funzioni più importanti del BMS è quella di monitorare parametri come la tensione, la temperatura e la corrente, sia dell'intera batteria che della singola cella, in modo da individuare eventuali malfunzionamenti, fornire un allarme e risolvere il problema senza danni.

Un'altra problematica che si deve affrontare nel caso di celle collegate in serie è quella dello sbilanciamento. Infatti, la differenza della capacità nominale tra le varie celle, dovuta ad errori di matching durante il processo costruttivo, provoca una degradazione della distribuzione della carica tra le celle. Tra le altre cause di sbilanciamento troviamo anche la differenza delle resistenze interne, della degradazione chimica, della temperatura ambiente e di quella interna delle celle durante la fase di carica e scarica [2]. Questo fenomeno può portare a situazioni in cui si ha nel pacco una cella completamente carica (o scarica) prima delle altre e questo causa un degrado delle prestazioni in quanto non viene più fornita (o prelevata) energia alla batteria. Il BMS si occupa anche di contrastare questo fenomeno, bilanciando la carica tra le celle. Questa funzionalità permette in sostanza di aumentare l'efficienza energetica della batteria e quindi la sua durata.

Un'altra funzione del BMS è quella di controllare la fase di carica, occupandosi di limitare la corrente per evitare il danneggiamento della batteria e di gestire una procedura che permetta di ottimizzare la carica totale ceduta alla batteria.

Per poter svolgere queste attività, il BMS deve conoscere lo stato di carica (State Of Charge, SOC) delle singole celle. Il SOC indica la carica residua presente in una cella, e viene espresso come percentuale della capacità nominale [3]. Lo stato di carica è una variabile di stato ed è ricavabile da parametri misurabili dal BMS, cioè dalla tensione e dalla corrente.

Un'altra variabile di stato molto importante è lo stato di salute (State Of Health, SOH) che riflette le condizioni in cui versa la batteria. Due indicatori del SOH sono la riduzione della capacità e l'incremento della resistenza interna della cella. Il BMS quindi si occupa

anche di quantificare e prevenire l'invecchiamento, cercando di evitare condizioni di stress che portano ad un peggioramento della capacità e della resistenza interna, rispetto al valore nominale.

Altre funzioni del BMS sono quelle di memorizzazione di alcune informazioni sulle celle (chimica, numero di serie, ecc.) e di permettere la comunicazione con l'utente, in modo da poter visualizzare lo stato della batteria o modificarne i parametri.

La comunicazione può avvenire anche verso un elaboratore nel caso in cui si vogliano raccogliere dati ed informazioni, utili per eventuali analisi. Solitamente viene utilizzata una comunicazione CAN, molto diffusa nell'ambiente automotive e industriale, grazie ad un'elevata immunità ai disturbi.

Riassumendo, il BMS è un sistema elettronico in grado di gestire una batteria, rilevando parametri di esercizio come la corrente, la tensione e la temperatura di ogni singola cella della batteria ed eseguendo operazioni di bilanciamento, protezione e ottimizzazione delle prestazioni [4].

1.3 Tipologie e architetture del BMS

Un BMS convenzionale è costituito da due parti principali: un'unità di controllo (Electronic Control Unit, ECU) ed un equalizzatore di carica [5]. La prima si occupa del calcolo dello stato di carica, del monitoraggio delle celle e della loro salvaguardia, mentre la seconda usa i dati ricavati dall'ECU per bilanciare le celle. Esistono diversi metodi di equalizzazione di carica e possono essere suddivisi in due grandi categorie: metodi passivi e attivi [6]. Le tecniche passive utilizzano dei resistori per dissipare l'energia in eccesso immagazzinata nelle celle più cariche. I metodi attivi invece

permettono di distribuire l'energia in eccesso verso le celle più scariche. La differenza sostanziale tra i due metodi è che quello passivo risulta essere meno efficiente rispetto a quello attivo, ma anche più semplice in termini di complessità circuitale e quindi meno costoso.

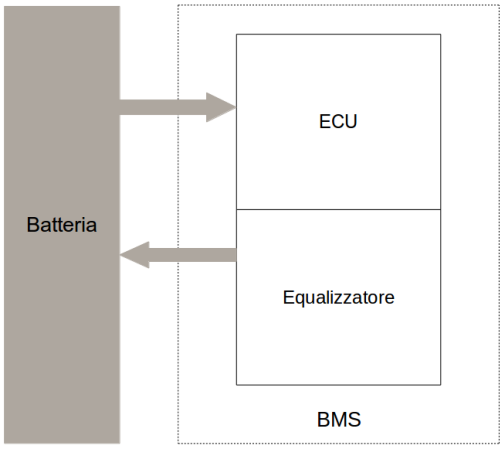

Figura 1: Architettura convenzionale di un BMS

Ultimamente si cerca di rendere quest'architettura più modulare per ottenere una maggiore flessibilità e ridurre l'ingombro dei collegamenti [5]. Si utilizzano quindi diverse unità locali (Local Electronic Control Unit, LECU) collegate ad un certo numero di celle (generalmente un segmento da 12 celle) e che comunicano tra loro e con un'unità centrale attraverso un bus di comunicazione (CAN-bus, SPI, RS-232, Ethernet, ecc.). L'unità centrale si occupa di calcolare il SOC, di gestire e sincronizzare i LECU, di garantire la sicurezza e di misurare la corrente del pacco. Visto che le celle sono connesse in serie basta un unico sensore per conoscere la corrente di ognuna di esse. Per evitare errori elevati sulla misura, i sensori di corrente devono essere accurati, soprattutto nell'intervallo di correnti di utilizzo, e devono essere a larga dinamica.

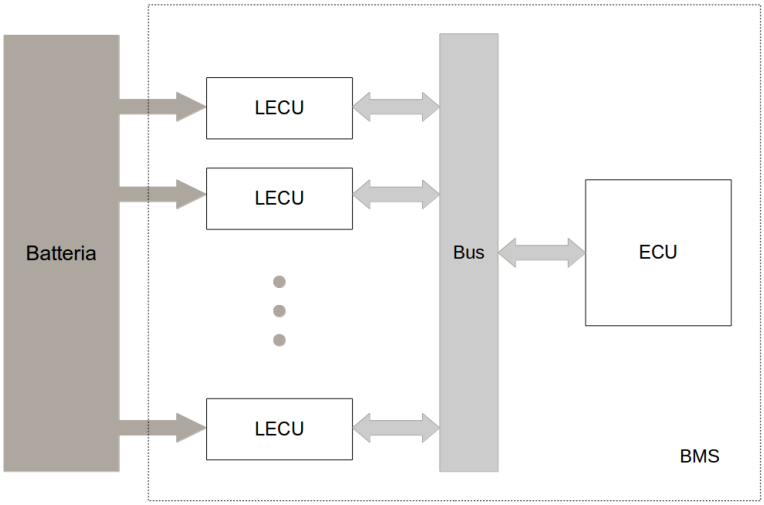

Figura 2: Architettura modulare di un BMS

Esistono diversi tipi di sensori per la misura della corrente. Un metodo è quello di utilizzare una resistenza di shunt, cioè un resistore di precisione con un valore molto basso, ai cui capi si ha una caduta di potenziale proporzionale alla corrente che ci scorre attraverso. Un altro sensore molto utilizzato è quello ad effetto Hall, che produce una tensione proporzionale ad una corrente elettrica, in presenza di un campo magnetico. A differenza dello shunt, permette di avere il segnale informativo isolato da quello di potenza, ma introduce alcuni effetti indesiderati, come ad esempio una forte dipendenza dalla temperatura che introduce del rumore nella misura. Un altro problema è l'errore di zero-offset, cioè il sensore non ha uscita pari a zero quando la corrente è nulla. Questo tipo di contributo può indurre un errore molto elevato nel calcolo dello stato di carica [7].

I LECU misurano la tensione delle singole celle e la temperatura del segmento e comunicano i dati all'unità centrale. La misura della tensione di cella deve essere eseguita con una certa accuratezza, soprattutto nell'intervallo di funzionamento sicuro,

in modo da evitare errori elevati nella stima dello stato di carica. Inoltre, occorrono dei sistemi in grado di prevenire eventuali errori di misura casuali, che potrebbero portare alla sotto-scarica o alla sovraccarica. In commercio esistono dei circuiti integrati, detti multicell battery stack monitor, che svolgono alcune delle funzionalità richieste ad un LECU. Essi possono misurare la singola tensione di cella, quella dell'intero modulo e la temperatura, garantiscono una maggiore accuratezza nella misura grazie alla possibilità di utilizzare una tensione di riferimento esterna [7] e hanno anche altre funzionalità ausiliarie.

2 Stima dello stato della batteria

In questo capitolo vengono descritti, e confrontati tra loro, alcuni degli algoritmi principali per la stima dello stato di carica. Alcuni di essi necessitano di un modello della cella e, per eseguire un calcolo accurato del SOC, è necessaria un'identificazione dei parametri del modello, durante la vita del sistema batteria. Dunque, viene fatta una panoramica sulla teoria dell'identificazione online dei parametri, presentando gli algoritmi più utilizzati.

2.1 Lo stato di carica (SOC)

Lo stato della batteria viene descritto da alcuni parametri che vengono ricavati utilizzando grandezze misurabili e modelli della batteria. Misure e modelli accurati contribuiscono ad una migliore stima dello stato.

Esistono diversi parametri interni, ognuno dei quali indica una determinata capacità o condizione della batteria. I più importanti sono lo stato di carica (SOC) e lo stato di salute (SOH), ma ne esistono diversi altri.

In alcune applicazioni, ad esempio, è importante conoscere lo stato di potenza (State Of Power, SOP), detto anche potenza istantanea disponibile, e lo stato di energia (State Of Energy, SOE). Questo accade nei veicoli elettrici, dove questi parametri possono essere necessari ad alcune unità di gestione dell'energia, per regolare l'emissione di energia di propulsione o la rigenerazione e l'immagazzinamento durante la frenata. Infatti, indicano se è possibile estrarre o fornire un certo valore di energia o potenza

costante, per un certo intervallo di tempo prefissato, senza eccedere i limiti di tensione e SOC imposti [8].

Il SOH serve a quantificare il grado di invecchiamento della batteria. Con l'utilizzo, la carica massima immagazzinabile nella cella diminuisce. L'invecchiamento dipende da vari fattori, come il numero di cicli di carica e scarica, la temperatura di funzionamento e a riposo, ecc.. Lo stato di salute può essere dunque definito nel seguente modo:

$$SOH(t) = \frac{Q_{max}(t)}{Q_n}$$

cioè come rapporto tra la capacità massima estraibile ad un certo istante ($Q_{max}(t)$) e la capacità nominale della cella (Q_n). Per determinare il SOH si possono utilizzare dei modelli d'invecchiamento chimico-fisici della cella oppure stimare i parametri circuitali di modelli elettrici tramite tecniche avanzate, come i filtri di Kalman o la stima nel senso dei minimi quadrati, per quantificare la riduzione di capacità e l'aumento della resistenza interna.

Il SOC indica la quantità di carica residua contenuta in una cella (Q_c), cioè la carica che può essere estratta dalla cella a temperatura ambiente e a C-rate bassi, dove C indica la corrente che scarica la cella in un'ora [4]. Lo stato di carica viene espresso come percentuale della capacità nominale:

$$SOC = \frac{Q_c}{Q_n}$$

Questa definizione può portare ad un errore nel caso in cui la capacità nominale sia inferiore a quella fornita dal costruttore, a causa dell'invecchiamento o di difetti costruttivi. Per eliminare questo inconveniente si potrebbe normalizzare il SOC con la carica effettivamente estraibile dalla cella. Il SOC è molto importante sia perché fornisce all'utilizzatore delle informazioni importanti, come il tempo residuo prima della scarica totale, sia perché in alcuni casi viene usato per stimare anche altri parametri.

2.2 Metodi di stima del SOC

Sono state realizzate diverse tecniche per la stima dello stato di carica, che si differenziano per complessità e precisione.

In applicazioni low power, come nei dispositivi portatili, possono essere utilizzati algoritmi più semplici in quanto non è richiesta una precisione elevata. Questo permette di contenere i costi. Al contrario, nelle applicazioni high power, come ad esempio nei veicoli elettrici, occorre una precisione ed un'affidabilità maggiore. Per questo motivo occorre usare tecniche più avanzate.

Di seguito vengono descritti e confrontati alcuni dei metodi di stima dello stato di carica più diffusi.

2.2.1 Coulomb Counting

Questa è la tecnica più utilizzata per la stima dello stato di carica ed è basata sull'integrazione della corrente nel tempo al fine di ottenere la carica immessa o estratta nella cella. Si ha dunque che la carica estratta dall'istante iniziale all'istante T è

$$Q = \int_0^T i_L(t)\,dt$$

dove $i_L(t)$ è la corrente misurata. La variazione dello stato di carica tra l'istante iniziale e il tempo *t* viene calcolata rapportando il valore di *Q* ad un valore di capacità di riferimento, scelto in base alla definizione di SOC usata (ad esempio la capacità nominale, Q_n):

$$SOC(t) = SOC_{init} - \eta_i \int_0^t \frac{i_L(\tau)}{Q_n} d\tau$$

dove SOC_{init} è il valore del SOC iniziale ed η_i è la coulombic efficiency, che serve a tenere conto della diversa efficienza di trasferimento della carica nelle fasi di carica e scarica ($\eta_i = 1$ in scarica e $\eta_i \leq 1$ in carica). Per le celle al litio-polimero questo coefficiente viene trascurato perché è circa pari ad 1 in entrambi le fasi. I risultati che si ottengono sono molto accurati, ma ci sono degli svantaggi:

- il valore del SOC iniziale deve essere preciso e non è ricavabile tramite integrazione, ma solo da una stima basata sulla OCV che, solitamente, non è molto precisa;
- la corrente che viene misurata è quella che fluisce dalla batteria verso il carico e non tiene conto delle correnti di auto-scarica;
- la coulombic efficiency è un parametro che dipende dallo stato della batteria (SOC, temperatura, ecc.) e questo influenza l'accuratezza del calcolo;
- questa tecnica non tiene conto di un eventuale rumore e offset sulla misura della corrente e questo si traduce in un errore sul SOC. Infatti, integrando ad anello

aperto un errore costante nel tempo, ad esempio un offset, questo si accumulerà nel tempo e farà divergere il valore dell'integrale.

Queste problematiche possono essere risolte, quando possibile, solo con una compensazione o una calibrazione periodica, effettuata in uno stato noto della batteria, come ad esempio quello di batteria completamente carica.

2.2.2 Discharge test

Questo test consiste in una scarica controllata della batteria, a corrente costante, e in una sua successiva ricarica. Questo è il metodo più affidabile per effettuare la stima, ma ha il grosso svantaggio di non essere utilizzabile in applicazioni online, a causa delle sua durata e del fatto che occorre una corrente costante di scarica e quindi bisognerebbe scollegare la batteria dal carico durante il test.

2.2.3 Open circuit voltage

Si può risalire allo stato di carica di una batteria osservando la sua tensione a vuoto (Open Circuit Voltage, OCV). Infatti, la tensione a vuoto è legata al SOC con una relazione non lineare che può essere descritta da una serie di coefficienti, ricavabili per via sperimentale. In generale, la OCV decrementa al diminuire del SOC. Per poter misurare la OCV bisogna attendere un tempo di rilassamento dell'ordine di alcune ore e quindi neanche questo metodo può essere utilizzato per applicazioni online.

Per alcuni tipi di batterie, come ad esempio per le LFP, la curva OCV-SOC risulta abbastanza piatta per un ampio intervallo di valori del SOC (Figura 3). Questa caratteristica può causare un grave errore sul SOC stimato nel caso in cui non si misuri

accuratamente la OCV.

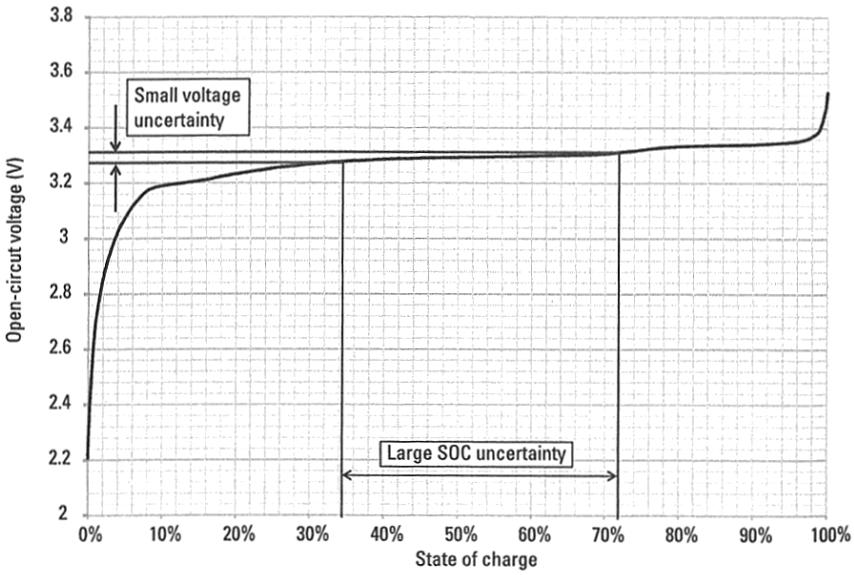

Figura 3: Caratteristica OCV-SOC per una cella LFP [7].

Inoltre, per SOC bassi la curva tende a scendere in maniera più ripida. Questo vuol dire che la relazione diventa fortemente non-lineare ed implica un aumento significativo dell'errore.

Altri problemi sono la dipendenza della curva OCV-SOC dalla temperatura (la tensione aumenta lievemente all'aumentare della temperatura e diminuisce nel caso contrario) ed un fenomeno di isteresi, visibile soprattutto nelle batterie LFP.

Alcuni algoritmi che utilizzano questa tecnica sono presentati in [9].

2.2.4 Neural Network

Una rete neurale è una rete costituita da più nodi (neuroni) che lavorano in parallelo e comunicano tra di loro al fine di trattare numerosi dati e approssimare funzioni non lineari, anche molto complesse. I nodi possono essere programmi o hardware dedicato. Come ingressi alla rete vengono forniti i valori misurati dalla batteria e altri coefficienti

con lo scopo di calcolare lo stato di carica, ottenendo un risultato molto accurato, indipendentemente dal tipo di batteria [10][11][12].

Per realizzare una stima di questo tipo occorre un elevato numero di dati storici ed un'elevata complessità computazionale. Queste caratteristiche rendono questo metodo non utilizzabile in molte applicazioni a causa del costo e dei consumi elevati.

2.2.5 Model based

Uno dei metodi più efficaci per la stima online del SOC è l'utilizzo di un modello della cella che ne simuli il funzionamento. Più il modello è accurato e più accurata è la stima effettuata a discapito di una maggiore complessità computazionale. Esistono diverse tipologie di modelli:

- modello elettrochimico: descrive i meccanismi fondamentali di generazione di potenza, legando informazioni macroscopiche (tensione e corrente) con quelle microscopiche (scambio di elettroni durante le reazioni di ossido-riduzione). Questi modelli sono complessi e costosi in termini di elaborazione e quindi non vengono impiegati nei sistemi di controllo, ma sono molto utili in fase di progettazione della cella, dove non si hanno costrizioni temporali e si necessita di una elevata precisione;
- modello matematico: caratterizza la batteria da un punto di vista quantitativo usando equazioni empiriche e metodi matematici. Modelli di questo tipo sono molto astratti, privi di significato fisico, e quindi vengono accoppiati ad altri modelli o utilizzati in applicazioni specifiche;
- modello circuitale: usa circuiti equivalenti a parametri concentrati per descrivere il

comportamento dell'accumulatore. Solitamente, consentono di avere tempi di calcolo brevi e una maggiore semplicità, a fronte di un'accuratezza inferiore.

La tipologia di modello da utilizzare dipende dall'applicazione. Nel caso di sistemi per la gestione del pacco batteria, i modelli più utilizzati sono quelli circuitali perché, oltre ad essere più veloci e semplici rispetto agli altri, hanno anche il vantaggio di essere più intuitivi e facili da maneggiare per i progettisti. In letteratura esistono diversi circuiti equivalenti che hanno in comune il fatto di poter essere suddivisi in tre parti fondamentali: una per modellare la tensione a vuoto, una per la resistenza interna e una per il transitorio presentato la cella.

In questa tesi viene utilizzato il modello in Figura 4 [13]. La parte sinistra del circuito modella la capacità della cella ed il SOC. A destra, il generatore controllato V_{OC} viene usato per simulare la relazione non lineare tra OCV e SOC, la resistenza R_0 rappresenta la resistenza interna della cella e la rete RC modella gli effetti di rilassamento. Collegando in serie più gruppi RC si può ottenere un modello più accurato.

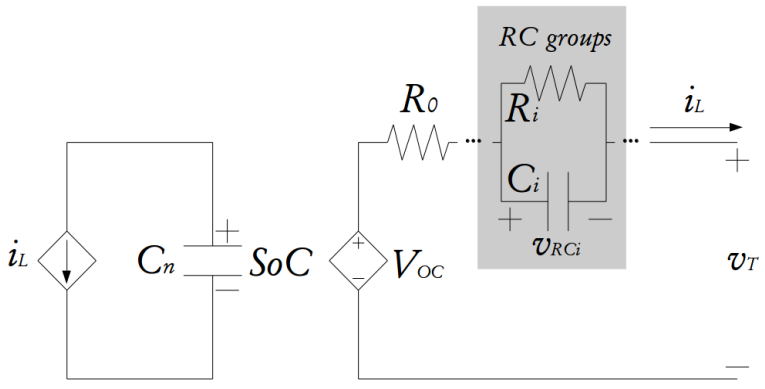

Figura 4: Circuito elettrico equivalente della cella.

La tensione v_T si riferisce a quella misurata ai capi della cella ed è formata da tre componenti:

$$v_T = V_{OC}(SOC, T) - V_{R_0}(SOC, T, i_L) - V_{RC}(SOC, T, i_L)$$

Dunque, conoscendo il valore della corrente di carico i_L e della tensione v_T , si può ricavare il valore di V_{OC} anche in condizioni dinamiche della batteria (quando la corrente di batteria si annulla, v_T rilassa verso V_{OC} con tempi nell'ordine delle ore). Dalla tensione a circuito aperto è possibile ricavare il valore dello stato di carica tramite la caratteristica OCV-SOC.

Per poter effettuare la stima del SOC usando un modello della cella bisogna ricorrere a degli osservatori, come ad esempio il Particle filter, il Kalman filter o un Mixed algorithm.

Un algoritmo misto prende il nome dal fatto che può essere visto come combinazione tra il metodo OCV e il Coulomb Counting, utilizzati insieme ad un modello. Una proposta di questo tipo sarà descritta successivamente.

Il Particle filter (PF) o Sequential Monte Carlo (SMC) è uno stimatore probabilistico con elevata precisione, utilizzato per stimare lo stato di un modello complesso non lineare. Questo algoritmo approssima la distribuzione a posteriori di un sistema, senza alcuna assunzione sulla forma della distribuzione, utilizzando dei campioni ottenuti con il metodo di Monte Carlo, ad ognuno dei quali viene associato un peso. Algoritmi che utilizzano un PF sono stati sviluppati in [14][15].

Il filtro di Kalman viene impiegato in molti campi (posizionamento globale,

comunicazioni, navigazione, ecc.) e ultimamente anche nel campo delle batterie, per stimarne lo stato.

Il filtro di Kalman è un metodo adatto alla stima del valore dello stato presente di un sistema dinamico. Quando alcune condizioni sono soddisfatte il filtro minimizza la covarianza d'errore stimata e, in questo caso, il filtro è ottimo [8]. Sostanzialmente, il filtro è costituito da un insieme di disequazioni nel dominio tempo-discreto, che vengono eseguite in modo ricorsivo, in tempo reale, le cui variabili modellano lo stato del sistema. Ad esempio, per una batteria potremmo avere come variabili il SOC, il SOH, ecc.. Ad ogni ricorsione, il filtro acquisisce gli ingressi e li utilizza, insieme ai valori dell'istante precedente, per calcolare le uscite che vengono poi confrontate con quelle reali del sistema. In questo modo viene ricavato il valore dell'errore d'uscita commesso che viene poi utilizzato per adattare lo stato del modello ed approssimarlo, in maniera più precisa, a quello del sistema reale. Per ottenere una stima accurata bisogna considerare nel modello quanti più fattori è possibile (dipendenza dalla temperatura, isteresi, ecc.), aumentandone quindi la complessità.

Se si deve stimare lo stato di un sistema non lineare bisogna utilizzare un processo di linearizzazione per ciascun istante di tempo, in modo da approssimarlo ad un sistema lineare tempo variante (LTV). Utilizzando il sistema così ottenuto insieme al Kalman Filter si ottiene il cosidetto Extended Kalman Filter (EKF).

L'Extended Kalman Filter viene usato negli algoritmi di stima presentati in [16] [17] [18] e [19].

2.2.6 Mixed Algorithm

Un algoritmo misto utilizza due o più metodi per realizzare una tecnica che abbia i vantaggi di entrambi.

Esistono diversi tipi di algoritmi misti, ma in questo elaborato viene trattato quello che utilizza il Coulomb Counting e il modello circuitale visto nel precedente paragrafo [20]. Questo algoritmo si presta bene ad essere utilizzato nei veicoli elettrici ed ibridi. Infatti, il Coulomb Counting è in grado di catturare bene i rapidi cambiamenti del SOC, ma, come detto in precedenza, è sensibile agli errori di misura sulla corrente. Quindi il modello circuitale viene usato per correggere dinamicamente il valore di SOC stimato dal Coulomb Counting e per seguire le variazioni del SOC di dinamica lenta.

2.3 Confronto tra i vari metodi

Esistono dunque diverse tecniche per la stima dello stato di carica ed ognuna di esse presenta vantaggi e svantaggi, in base ai quali queste tecniche sono idonee ad essere utilizzate in un'applicazione anziché in un'altra.

Nella tabella seguente vengono messi a confronto i metodi trattati.

Metodo	Applicazione	Vantaggi	Svantaggi
Coulomb Counting	Qualsiasi tipo di sistema	Accuratezza, facilità d'implementazione, stima online	Calibrazione periodica, dipendenza dal valore iniziale del SOC, sensibile al rumore e all'offset di misura
Discharge test	Misura della capacità di una cella all'inizio del proprio ciclo di vita	Facilità di utilizzo, accuratezza	Solo per applicazioni offline, durata, perdita di energia
Open circuit voltage	In sistemi che permettono di avere lunghi tempi di attesa	Utilizzabile per ogni tipo di batteria, utilizza poche risorse	Le misure devono essere effettuate a corrente nulla dopo una lunga pausa
Neural Network	Qualsiasi tipo di sistema	Stima online, accuratezza	Necessita di uno storico di dati, costo d'implementazione
Filtri di Kalman	Qualsiasi tipo di sistema, anche molto dinamico (HEV)	Stima online, accuratezza, flessibilità	Forti ipotesi sul modello, difficoltà d'implementazione, risorse
Mixed Algorithm	Qualsiasi tipo di sistema, anche molto dinamico (HEV)	Stima online, facilità d'implementazione del Coulomb Counting senza i suoi svantaggi	L'accuratezza della stima dipende da quella del modello circuitale usato

Tabella 2: Confronto tra i metodi di stima del SOC

2.4 L'identificazione online dei parametri nei modelli

Dalle considerazioni fatte nei precedenti paragrafi, risulta evidente che i metodi più adatti a stimare lo stato di carica, in applicazioni su veicoli elettrici ed ibridi, sono il Kalman Filter (in particolare la sua variante per sistemi non lineari, cioè l'Extended Kalman Filter) e il Mixed Algorithm. Il Battery Management System deve essere in grado di simulare il comportamento della batteria, utilizzando uno di questi due modelli,

e di confrontare i risultati con i valori reali misurati, per poi agire di conseguenza.

Per soddisfare i requisiti di accuratezza richiesti in campo automotive non basta avere un buon modello della cella.

Infatti, visto che i parametri di una batteria non sono costanti nel tempo, ma cambiano a causa dell'invecchiamento e dipendono dal SOC e dalla temperatura, occorre un modello in grado di aggiornare i propri parametri, tramite l'identificazione online, in modo che la stima rispecchi sempre la dinamica del sistema.

In generale, un sistema è un oggetto in cui variabili di natura diversa (ingressi e disturbi) interagiscono e producono segnali osservabili (uscite). La conoscenza delle modalità d'interazione tra le variabili è il modello del sistema. L'obiettivo dell'identificazione è quello di determinare la struttura "ottimale" del modello matematico e dei coefficienti (parametri), presenti nelle equazioni del modello, a partire dall'osservazione dei dati.

Una classificazione molto importante sull'identificazione riguarda il fatto che avvenga online, cioè durante il funzionamento del sistema, oppure offline. Nell'identificazione online si utilizzano le informazioni per regolare le strategie di controllo che vengono implementate sul segnale, in modo adattivo rispetto agli errori di stima sui parametri o alla deriva degli stessi [5].

2.5 Modelli per l'identificazione

Per poter risolvere il problema dell'identificazione occorre una certa quantità di dati. Questi dati vengono utilizzati per ricavare le relazioni che intercorrono tra una certa variabile e gli altri dati disponibili.

La relazione più semplice possibile è quella lineare. Modelli basati su una relazione

lineare si dicono regressivi e sono descritti da una regressione lineare:

$$y(k)=\varphi_1\vartheta_1+\varphi_2\vartheta_2+...+\varphi_N\vartheta_N$$

dove *y* è la variabile dipendente esprimibile come relazione lineare, secondo i coefficienti ϑ_i, delle variabili indipendenti φ_i, dette regressori. L'identificazione si effettua individuando il valore dei parametri incogniti ϑ_i. Inoltre, avendo a che fare con dati reali, non si sa se la relazione lineare sussista o meno, ma la si può comunque supporre verificata per ogni campione k e scrivere

$$y(k)=\varphi_1(k)\vartheta_1+\varphi_2(k)\vartheta_2+...+\varphi_N(k)\vartheta_N+e(k)=\varphi(k)^T\vartheta+e(k)$$

dove $e(k)$ è detto errore di equazione ed è l'errore che si commette considerando che la relazione di regressione lineare sia esatta.

Il modello regressivo è valido solo se si ha a disposizione una raccolta di dati tale che il k-esimo campione della variabile *y* sia legato linearmente ai k-esimi campioni delle variabili φ_i. Spesso capita però di avere i campioni ordinati secondo una sequenza temporale, come accade per esempio per le serie storiche, ossia sequenze ordinate di variabili osservate ad intervalli regolari. In questo caso un modello può essere costruito basandosi sul fatto che queste sequenze presentano una certa struttura interna (variazione stagionale, autocorrelazione, ecc.). Dunque è possibile che il valore attuale di $y(k)$ dipenda dai valori precedenti secondo la relazione

$$y(k)=a_1 y(k-1)+a_2 y(k-2)+...+a_N y(k-N)=\sum_{i=1}^{N} a_i y(k-i)$$

detta modello autoregressivo o AR (Auto Regressive).

Nei modelli autoregressivi spesso il valore di *y*, in un certo istante, dipende anche da un'altra variabile indipendente, detta ingresso esogeno, indicata con *u*. Si ha quindi che

$$y(k)=\sum_{i=1}^{N_a} a_i y(k-i)+\sum_{i=0}^{N_b} b_i u(k-i-d)$$

dove *d* indica il minimo ritardo tra ingresso e uscita. Un modello di questo tipo si dice autoregressivo esogeno o modello ARX (Auto Regressive eXogenous). L'equazione che lo descrive viene chiamata, in ambito sistemistico, equazione alle differenze e, nel caso in cui tutti i coefficienti a_i sono nulli, il modello si chiama FIR (Finite Impulse Response). Anche nei modelli autoregressivi si può supporre esatta la relazione considerando un errore additivo. In questo caso si parla di modello ad errore di equazione [5].

2.6 Metodo dei minimi quadrati

I metodi di stima online dei parametri sviluppati possono essere classificati in due categorie: i metodi basati sul filtro di Kalman e quelli basati sui minimi quadrati [7] [21]. Generalmente, i modelli basati sul filtro di Kalman forniscono una soluzione accurata, ma in alcuni casi questo non accade. Infatti, si ottiene un grosso errore quando l'errore di misura e di processo sono incorrelati, gaussiani bianchi a media nulla e la loro covarianza non è definita correttamente. Inoltre, l'elevata complessità richiesta ne rende

difficoltosa l'implementazione in sistemi embedded real-time.

I metodi basati sui minimi quadrati (Least Squares, LS) sono ampiamente utilizzati per la stima dei parametri grazie al loro basso costo computazionale e all'accuratezza relativamente alta. Esistono diverse varianti del metodo, come i minimi quadrati pesati, i minimi quadrati ricorsivi (Recursive Least Square, RLS) e il Moving Window Least Square (MWLS).

Il metodo dei minimi quadrati è una tecnica che permette di ricavare una soluzione approssimata di sistemi sovra-determinati, cioè composti da un numero di equazioni maggiore rispetto alle variabili da determinare. Il nome deriva dal fatto che per ricavare questa soluzione viene minimizzato l'errore quadratico medio di ogni equazione.

Lo scopo di questa tecnica è quello di aggiustare i parametri del modello di un sistema reale in modo da approssimare al meglio una collezione di dati. Quando le variabili possono essere modellate come combinazione lineari di altri dati osservati si ha:

$$y(k)=\varphi_1(k)\vartheta_1+\varphi_2(k)\vartheta_2+...+\varphi_{N_p}(k)\vartheta_{N_p}=\varphi(k)^T\vartheta$$

dove $\varphi(k)$ sono le variabili indipendenti e $y(k)$ quelle dipendenti. Quindi, considerando N campioni, la regressione lineare appena descritta può essere messa nella forma compatta:

$$y(N)=\Phi(N)\vartheta$$

L'identificazione tramite LS permette di ricavare il vettore dei parametri ϑ a partire dagli N campioni di *y* e dal vettore dei regressori φ. L'equazione in forma compatta

descrive un sistema di N equazioni lineari in N_p incognite (vettore ϑ) e, se N_p>N, il sistema è sovra-determinato. In questo caso, la soluzione ottima è quella che minimizza la sommatoria dei residui elevati al quadrato:

$$S=\sum_{i=1}^{n} r_i^2$$

Il residuo si definisce come la differenza tra il valore attuale della variabile dipendente e del valore calcolato dal modello:

$$r_i=f(\varphi_i,\vartheta_i)-y_i$$

Talvolta, i dati acquisiti non hanno tutti la stessa importanza e quindi si necessita di un modo per considerarli con una minore importanza. Questo comportamento si può ottenere introducendo il fattore di oblio (forgetting factor), un peso tanto più elevato quanto maggiore è l'importanza che si vuole dare al campione. Questo metodo prende il nome di minimi quadrati pesati [21].

Per eseguire un'identificazione online non possono essere utilizzati i metodi appena presentati in quanto funzionano a lotti, ossia lavorano su un gruppo di dati per ottenere un'unica stima. Per applicazioni in linea si ricorre all'algoritmo dei minimi quadrati ricorsivi che aggiorna i parametri secondo una legge che permette di ricavare la stima in un istante basandosi su quella dell'istante precedente e sugli ultimi dati acquisiti.

Un'altra tecnica che permette di eseguire un'identificazione online è la MWLS, metodo utilizzato in questo elaborato e che vedremo nel dettaglio nei prossimi capitoli.

2.7 Algoritmi per la risoluzione dei problemi di minimi quadrati

Come detto nel paragrafo precedente, il metodo LS serve a ricavare una soluzione approssimata di sistemi sovra-determinati. Quindi, generalizzando il problema, si vuole ricavare il vettore $x \in \mathbb{R}^n$ tale che $Ax = b$, dove $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$ sono noti e $m \geq n$. La matrice *A*, essendo rettangolare, non ammette inversa e quindi il sistema non ha una soluzione generale, ma solo soluzioni che minimizzano il residuo sotto una particolare metrica. Il metodo dei minimi quadrati cerca di ricavare il vettore *x* minimizzando la quantità $\|Ax - b\|_2$.

Esistono diversi metodi per la ricerca di questa soluzione approssimata. In particolare, due algoritmi di fattorizzazione sono stati descritti in [22] e poi confrontati in base alla loro implementabilità su FPGA e alla complessità in termini di flops (floating point operations): la fattorizzazione QR e la fattorizzazione Cholesky.

2.7.1 Fattorizzazione QR

La fattorizzazione QR (QR Decomposition, QRD), consiste nello scomporre la matrice *A* in una matrice ortonormale $Q \in \mathbb{R}^{m \times m}$ e una matrice $R \in \mathbb{R}^{m \times n}$ tale che la sua sottomatrice *n x n* più in alto sia triangolare superiore [23]. Si ottiene dunque:

$$A = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix} \Rightarrow Q^T A = R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \begin{matrix} n \\ m-n \end{matrix}$$

Risolvere un sistema nel senso dei minimi quadrati vuol dire ricavare il vettore *x* che minimizzi la quantità $\|Ax - b\|_2$. Si ha quindi che:

$$\|Ax-b\|_2=\|QRx-b\|_2=\|Q\left(Rx-Q^T b\right)\|_2=\|Rx-Q^T b\|_2$$

e ponendo

$$Q^T b=\begin{bmatrix} d \\ c \end{bmatrix} \begin{matrix} n \\ m-n \end{matrix}$$

si può scrivere che

$$\|Rx-Q^T b\|_2= \left\|\begin{bmatrix} R_1 \\ 0 \end{bmatrix} x-\begin{bmatrix} d \\ c \end{bmatrix}\right\|_2 = \left\|\begin{bmatrix} R_1 x-d \\ -c \end{bmatrix}\right\|_2 = \|c\|_2+\|R_1 x-d\|_2$$

Il minimo di $\|Ax-b\|_2$ è $\|c\|_2$ e si ha quando $R_1 x-d=0$, da cui il vettore *x* tale che $R_1 x=d$ è la soluzione nel senso dei minimi quadrati.

Per applicare la decomposizione QR si possono usare diversi algoritmi, ma i due più diffusi sono l'algoritmo di Gram-Schmidt e l'algoritmo con Givens rotation. L'algoritmo di Gram-Schmidt ottiene le basi ortogonali scandendo tutte le colonne della matrice *A* da decomporre, mentre, per derivare la matrice triangolare superiore, usa il principio di ortogonalità. Invece, La rotazione di Givens (GR) azzera un elemento alla volta, in modo iterativo, mediante una rotazione bi-dimensionale [24].

In tutti i casi, dopo aver eseguito la decomposizione, si dovrà poi ricavare il vettore *x*:

$$Ax=QRx=b \;\Rightarrow\; Rx=Q^T b \;\rightarrow\; \begin{cases} d=Q^T b \\ Rx=d \end{cases}$$

si effettuerà quindi una sostituzione diretta per calcolare $d=Q^T b$ e una sostituzione inversa per ricavare x dal sistema $Rx=d$.

2.7.2 Fattorizzazione Cholesky

Si supponga di avere un sistema $Bx=y$ dove $B\in\mathbb{R}^{m\times n}$, $x\in\mathbb{R}^n$, $y\in\mathbb{R}^m$ e $m\geq n$ (sistema sovra-determinato). Il problema della risoluzione nel senso dei minimi quadrati può essere trasformato nel seguente modo:

$$\underbrace{B^T B}_{A}x=\underbrace{B^T y}_{b}\rightarrow Ax=b$$

dove $A\in\mathbb{R}^{n\times n}$ è una matrice simmetrica e semi-definita positiva (cioè $(Av)\cdot v\geq 0$ $\forall v\in\mathbb{R}^n$) e $b\in\mathbb{R}^n$. Fattorizzando *A* in modo opportuno, il calcolo di *x* (soluzione nel senso dei minimi quadrati) risulta immediato, senza effettuare l'inversione della matrice *A*. L'algoritmo di Cholesky presenta diverse varianti. La fattorizzazione LL^T fattorizza la matrice *A* nel prodotto di due matrici triangolari, $A=L\,L^T$, con *L* triangolare inferiore e, di conseguenza, L^T triangolare superiore. Si ottiene dunque il seguente sistema:

$$Ax=L\,L^T x=b\ \rightarrow\ \begin{cases} Ly=b \\ L^T x=y \end{cases}$$

e si procede, come nel caso precedente, ad una sostituzione diretta per ricavare *y* da $Ly=b$ e poi ad una sostituzione inversa per ottenere *x* dal sistema $L^T x=y$.

Questo algoritmo può essere semplificato, eliminando il calcolo di alcune radici quadrate presenti nella fattorizzazione, così da rendere la risoluzione del sistema meno complessa e più veloce. Si può dunque usare la fattorizzazione Cholesky modificata (LDL^T), che consiste nel fattorizzare *A* come $A=LDL^T$ dove *D* è una matrice diagonale e *L* è una matrice triangolare inferiore con gli elementi della diagonale unitari. Si passa quindi alle sostituzioni:

$$Ax=b \;\Rightarrow\; LDL^T x=b \;\rightarrow\; \begin{cases} Lz=b \\ Dh=z \\ L^T x=h \end{cases}$$

risolvendo in *z* la prima equazione, in *h* la seconda ed infine in *x* la terza.

2.7.3 Confronto tra gli algoritmi trattati

Per scegliere quale algoritmo utilizzare occorre eseguire un confronto in termini di complessità e prestazioni tra i vari metodi. Un'analisi di questo tipo si trova in [22] e in [23] e viene riassunta nella Tabella 3.

Algoritmo	Flops	Radici quadrate
QRD con Gram-Schimdt	$2mn^2$	n
QRD con Givens rotation	$2(n^2m-n^3/3)$	$mn - n^2/2$
Cholesky (LL^T)	$n^3/3$	n
Cholesky modificata (LDL^T)	$n^3/3$	0

Tabella 3: Confronto tra i principali algoritmi di risoluzione di un sistema del tipo Ax=b nel senso dei minimi quadrati, dove A è una matrice di m righe ed n colonne e b è un vettore di m elementi.

Si può osservare che l'algoritmo di decomposizione QR più conveniente, in termini di flops, è quello con Givens rotation, ma, per determinati valori di *m* e *n,* può aver bisogno di più radici quadrate da calcolare. Questo algoritmo ha anche il vantaggio di poter essere implementato utilizzando l'algoritmo CORDIC (Coordinate Rotation Digital Computer). Il CORDIC è un algoritmo iterativo che permette di ruotare vettori mediante solo operazioni elementari (somme algebriche e shift) e quindi è facilmente descrivibile in HDL ed è facilmente implementabile su FPGA.

Per quanto riguarda la fattorizzazione Cholesky, la più conveniente è quella modificata,

in quanto non necessita di calcolare nessuna radice quadrata e presenta il vantaggio di poter aggiungere uno stadio di pipeline in più, rispetto all'algoritmo LL^T, aumentando il throughput e, di conseguenza, le prestazioni del sistema. La fattorizzazione LDL^T, in termini di flops, è meno complessa di tutti gli altri algoritmi. Non va però dimenticato che, prima della fattorizzazione, vanno effettuati dei calcoli matriciali per ottenere B^TB, per cui sono necessari n^2m flops, e per ottenere B^Ty, *per cui* occorrono nm flops. Inoltre bisogna considerare che anche i passaggi successivi di sostituzione diretta e inversa hanno complessità diverse in termini di flops.

Quindi, in conclusione, la fattorizzazione Cholesky e la decomposizione QR con Givens rotation sono circa equivalenti: la scelta dovrà essere basata sul tipo di applicazione e sulle dimensioni delle matrici.

2.8 Stato dell'arte sulla realizzazione degli algoritmi di stima

Attualmente, tutte le funzionalità del BMS, compreso il calcolo dello stato di carica, vengono implementate su un microcontrollore, in modo da semplificare lo sviluppo del sistema e di ridurne i costi.

In particolare, quando non si necessita di un'elevata precisione e affidabilità, si possono utilizzare algoritmi di stima molto semplici e modelli con parametri fissi. In questo modo sono necessarie poche risorse di calcolo e l'intero sistema può essere implementato su un unico microcontrollore.

Per alcune applicazioni questa soluzione non è realizzabile, come ad esempio nelle applicazioni automotive, in cui sono necessarie una precisione ed un'affidabilità più

elevate e quindi risorse di calcolo maggiori. Per questo motivo occorrono BMS che utilizzano algoritmi di stima più precisi, e quindi più complessi, e che usano modelli con identificazione online dei parametri (per effettuare la stima online occorre elaborare matrici di grandi dimensioni ed effettuare operazioni piuttosto complesse). Servono inoltre funzioni che permettano di evitare e gestire al meglio gli errori casuali e di gestire numerose celle connesse tra loro. Si utilizzano quindi microcontrollori con prestazioni più elevate e strutture modulari, come descritto nel paragrafo 1.3, in cui alcune funzioni del BMS vengono distribuite tra i vari nodi di calcolo del sistema.

Algoritmi così complessi, che lavorano su grosse matrici si possono implementare con risultati migliori, in termini di prestazioni, su FPGA. Si possono così realizzare degli acceleratori hardware che lavorano in parallelo al microcontrollore e che si occupano di calcolare il SOC di tutte le celle presenti nella batteria e di comunicarlo all'unità centrale. Un'altra soluzione è quello di creare un SoPC (System on a Programmable Chip), implementando la stima del SOC e l'identificazione dei parametri come periferiche o come coprocessori hardware del processore principale.

Tuttora, sistemi di questo tipo, non sono presenti in commercio e sono poco diffusi anche in letteratura. Un lavoro in questo senso è presentato in [25], dove è stato sviluppato un algoritmo di stima che usa il Kalman filter su FPGA.

3 Algoritmo di stima proposto

Dalle riflessioni effettuate nel capitolo precedente emerge che gli algoritmi più idonei ad essere utilizzati nei veicoli elettrici ed ibridi sono quelli basati sul filtro di Kalman e gli algoritmi misti. In questo elaborato è stato sviluppato un algoritmo di stima del SoC basato su una variante del Mixed Algorithm, nel quale i parametri del modello di cella vengono costantemente aggiornati tramite una identificazione online realizzata mediante il metodo dei Minimi Quadrati.

L' algoritmo viene descritto nel dettaglio in questo capitolo. Vengono inoltre presentate delle analisi sulla stabilità del sistema e sul suo comportamento in presenza di errori, studio effettuato in [5].

3.1 Descrizione generale

L'algoritmo realizzato può essere suddiviso in due parti: la parte che si occupa della stima dello stato di carica e quella che realizza l'identificazione online dei parametri del modello di cella (Figura 5).

Il blocco di stima del SOC utilizza il Mixed Algorithm basato sugli algoritmi del Coulomb Counting e del Model based. Lo stato di carica viene calcolato dal modulo Coulomb Counting, dove però la corrente integrata è data da quella misurata e da un termine correttivo, proporzionale alla differenza tra la tensione di cella misurata e quella fornita dal modello.

La parte di identificazione online dei parametri utilizza gli ingressi misurati per calcolare

i parametri del modello tramite la tecnica del Moving Window Least Square (MWLS).

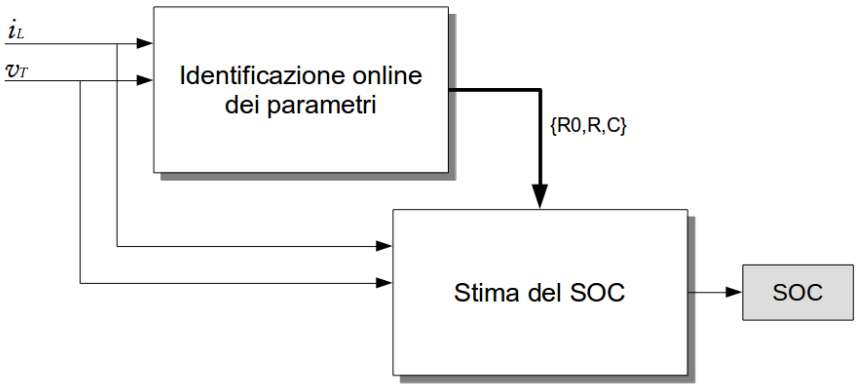

Figura 5: Struttura generale dell'algoritmo.

Durante il funzionamento del sistema le due parti comunicano tra di loro. Infatti, il blocco di stima del SOC utilizza i parametri calcolati dalla parte che esegue l'identificazione per aggiornare il proprio modello della cella.

3.2 Algoritmo di stima del SOC

Lo schema a blocchi complessivo dell'algoritmo viene presentato in Figura 6. Viene utilizzato il Mixed Algorithm che consiste nella stima dello stato di carica tramite la tecnica del Coulomb Counting, i cui problemi, descritti nel precedente capitolo, vengono risolti mediante un anello di retroazione che applica un termine correttivo alla corrente misurata i_L. Il termine correttivo viene calcolato con l'utilizzo di un modello della cella che ne simula il comportamento. La corrente così ottenuta viene inviata al blocco d'integrazione che ricava il valore del SOC dalla seguente relazione:

$$SOC(t) = SOC_{init} + \frac{1}{Q_R}\int_{t_0}^{t} i(\tau)\, d\tau$$

Il SOC calcolato viene inviato ad una LUT che contiene i punti della caratteristica OCV-SOC e che fornisce in uscita il valore della tensione a circuito aperto (V_{OC}) corrispondente. Questo valore, sommato alla tensione v_Z, rappresenta la stima della tensione della cella calcolata dal modello. I parametri del modello vengono aggiornati periodicamente con i valori provenienti dal blocco d'identificazione. Sottraendo al valore di tensione stimato quello reale misurato v_T, si ottiene l'errore di stima che viene moltiplicato per un opportuno coefficiente e poi sottratto alla corrente da integrare.

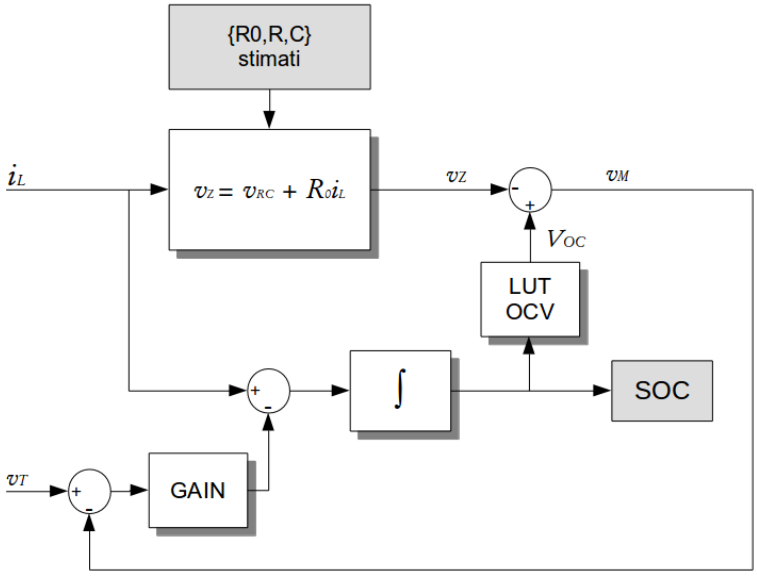

Figura 6: Schema a blocchi dell'algoritmo di stima del SOC.

Si ottiene dunque una stima più accurata rispetto all'uso del solo Coulomb Counting grazie all'utilizzo di una retroazione con un controllo di tipo proporzionale che permette di annullare gli eventuali errori.

3.2.1 Modello della batteria

Il modello della batteria utilizzato è già stato presentato nel precedente capitolo. In particolare si utilizza il modello con un gruppo RC, chiamato modello ridotto, rappresentato in Figura 7 (la variante con due gruppi RC viene indicata con il nome di modello completo).

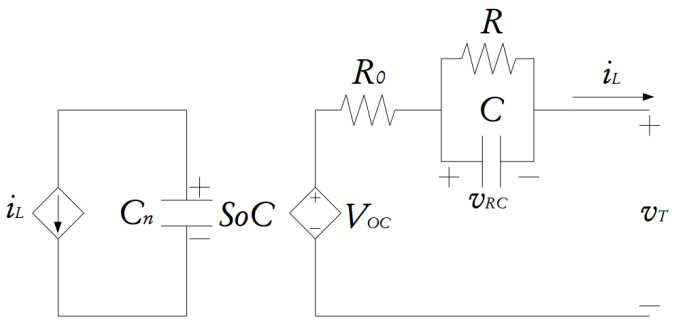

Figura 7: Modello circuitale ridotto.

L'utilizzo di un solo gruppo RC permette di ridurre molto la complessità del sistema, pur mantenendo una buona accuratezza.

Il generatore controllato V_{OC} modella la relazione non lineare tra stato di carica e tensione a circuito aperto ($V_{OC}=f(SOC)$). Questa relazione può essere espressa per mezzo di sistemi di equazioni che comportano una complessità maggiore del modello, soprattutto nel caso in cui considerano gli effetti di isteresi e di dipendenza dalla temperatura. Inoltre, la trasformazione di una curva sperimentale in un sistema di equazioni porta inevitabilmente ad un errore. Per questi motivi viene utilizzata una Look-up table (LUT OCV) contenente i valori della curva ottenuta sperimentalmente da test offline. Nonostante questa curva sia non lineare, visto che per i valori di corrente di carica e scarica ordinari il SOC subisce “piccole” variazioni, si può linearizzare la

relazione, approssimando la curva con una spezzata (sliding line) con pendenza variabile, b_1, e punto d'intersezione con l'asse V_{OC}, b_0:

$$V_{OC} = f(SOC) = b_0 + b_1 SOC$$

Le equazioni stato-spazio nel dominio del tempo che descrivono il modello sono riportate nel seguente sistema:

$$\begin{cases} \frac{d}{dt} SOC(t) = -\frac{i_L(t)}{Q_R} \\ \frac{d}{dt} v_{RC}(t) = -\frac{v_{RC}(t)}{RC} + \frac{i_L(t)}{C} \\ v_T(t) = V_{OC} - R_0 i_L(t) - v_{RC}(t) \end{cases}$$

Q_R è la carica totale estraibile dalla cella. Le variabili di stato del sistema sono il *SOC* e la v_{RC}, cioè la tensione ai capi del gruppo RC. La corrente i_L e la tensione v_T sono le uniche variabili accessibili del sistema in quanto direttamente misurabili.

3.3 Analisi della stabilità e della sensibilità agli errori

Come già anticipato, la tecnica del Coulomb Counting e quella del Model based, prese singolarmente, presentano degli svantaggi dovuti agli errori di misura e all'inizializzazione. In particolare la corrente viene misurata da un sensore di Hall che presenta un errore costituito da componenti ad alta frequenza, picchi e un offset a bassa frequenza. Anche la tensione presenta un certo errore causato dalla conversione

effettuata da un ADC. Questi errori tendono a far divergere il valore della stima del SOC calcolato dall'algoritmo del Coulomb Counting.

Di seguito viene riportata un'analisi per analizzare il comportamento dell'algoritmo utilizzato in presenza di errori e la sua stabilità.

3.3.1 Analisi sulla stabilità

L'analisi sulla stabilità del sistema viene effettuata nel dominio della frequenza e viene realizzata valutando l'effetto di ogni singolo ingresso (I_L e V_T), in accordo con il principio di sovrapposizione. L'algoritmo è asintoticamente stabile se la funzione di trasferimento di ciascun ingresso è asintoticamente stabile. Le funzioni di trasferimento possono essere ricavate facendo riferimento allo schema a blocchi in Figura 8, dove la tensione di cella viene calcolata utilizzando la relazione lineare della V_{OC} ricavata nel paragrafo precedente. Si ha quindi che la tensione della cella ricavata dal modello è pari a:

$$V_M = V_{OC} - Z_S(s) I_L = b_0 + b_1 SOC - Z_S(s) I_L$$

con $Z_S(s) = R_0 + \frac{R}{1+RCs}$. Il contributo di b_0 non viene preso in considerazione in un'analisi alle variazioni. Il controllo di tipo proporzionale viene realizzato con una moltiplicazione per una costante K_p. Il SOC calcolato dal blocco integratore, a meno dell'inizializzazione, è dato, nel dominio di Laplace, dalla seguente relazione:

$$SOC(s) = -\frac{I_L(s)}{sQ_R}$$

dove il segno negativo è dovuto al fatto che la corrente in carica è negativa, per convenzione.

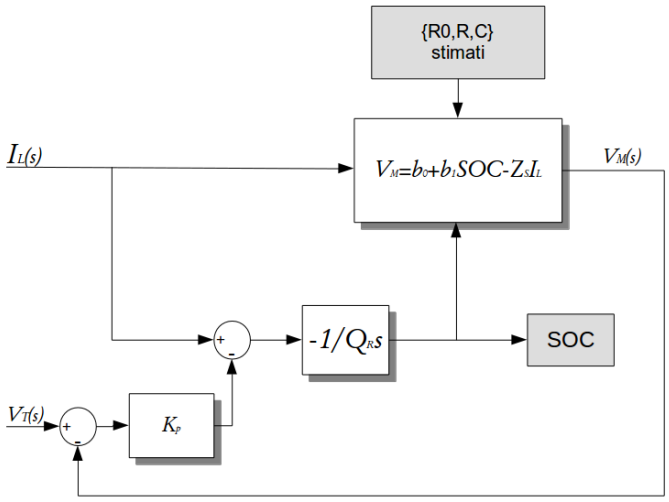

Figura 8: Schema a blocchi dell'algoritmo di stima del SOC, nel dominio di Laplace.

La funzione di trasferimento tra *SOC* e V_T si ricava ponendo l'ingresso I_L pari a zero:

$$SOC(s)\big|_{I_L=0}=\frac{K_p}{Q_R s}\left(V_T(s)-b_1 SOC(s)\right)$$

da cui

$$W_v(s)\equiv\frac{SOC(s)}{V_T(s)}=\frac{K_p}{Q_R}\frac{1}{s+\frac{K_p b_1}{Q_R}}$$

Per ottenere la funzione di trasferimento per l'ingresso I_L, bisogna considerare nullo l'ingresso V_T:

$$SOC(s)\big|_{V_T=0} = -\frac{1}{Q_R s}\left(I_L(s) + K_p b_1 SOC(s) - K_p Z_s(s) I_L(s)\right) = I_L(s)\left(\frac{K_p Z_s(s) - 1}{Q_R s}\right) - \frac{K_p b_1 SOC(s)}{Q_R s}$$

da cui

$$W_I(s) \equiv \frac{SOC(s)}{I_L(s)} = \frac{K_p Z_s(s) - 1}{Q_R} \frac{1}{s + \frac{K_p b_1}{Q_R}}$$

Le funzioni di trasferimento hanno lo stesso denominatore e quindi, in entrambi i casi, la stabilità dipende dal segno del termine $\frac{K_p b_1}{Q_R}$. In realtà, sia b_1 che Q_R sono positivi e quindi bisogna prendere in considerazione solo il segno di K_p. Se $K_p = 0$ la retroazione viene annullata e il sistema lavora ad anello aperto, cioè il SOC viene calcolato tramite il metodo del Coulomb Counting, mentre, se $K_p > 0$, il sistema risulta asintoticamente stabile.

3.3.2 Analisi sugli errori di misura

Per analizzare le prestazioni dell'algoritmo in presenza di errori di misura (rumore ed offset) sulle grandezze d'ingresso si utilizza un parametro chiamato errore di stima, definito come $SOC_{err} = SOC_{true} - SOC$, dove SOC_{true} è il valore reale dello stato di carica mentre SOC è quello stimato. In particolare, si vuole valutare come ciascun errore incida sull'errore di stima.

Per quanto riguarda gli errori dovuti a misurazioni di tensione si considera un errore costante V_{err} e si applica la sovrapposizione degli effetti per determinare la risposta del sistema ad un ingresso a gradino di ampiezza pari a V_{err}:

$$SOC_{err}(s)=\frac{V_{err}}{s}W_V(s)=\frac{V_{err}}{s}\frac{K_p}{Q_R}\frac{1}{s+\frac{K_p b_1}{Q_R}}$$

Utilizzando il teorema del valore finale si può determinare il valore di SOC_{err} a regime:

$$\lim_{s\to\infty} SOC_{err}(t)=\lim_{s\to 0} s\,SOC_{err}(s)=\lim_{s\to 0} s\frac{V_{err}}{s}\frac{K_p}{Q_R}\frac{1}{s+\frac{K_p b_1}{Q_R}}=\frac{V_{err}}{b_1}$$

Come si nota, il sistema non è in grado di correggere un errore sulla tensione. Inoltre, il risultato non dipende da K_p, ma solo da b_1 che è un parametro caratteristico della cella su cui non si può agire.

In presenza di un errore di misura sulla corrente si può applicare lo stesso procedimento:

$$SOC_{err}(s)=\frac{I_{err}}{s}W_I(s)=\frac{I_{err}}{s}\frac{K_p Z_s(s)-1}{Q_R}\frac{1}{s+\frac{K_p b_1}{Q_R}}$$

da cui, applicando il teorema del valore finale, si ottiene:

$$\lim_{s\to\infty} SOC_{err}(t)=\lim_{s\to 0} s\,SOC_{err}(s)=\lim_{s\to 0} s\frac{I_{err}}{s}\frac{K_p Z_s(s)-1}{Q_R}\frac{1}{s+\frac{K_p b_1}{Q_R}}=I_{err}\frac{K_p Z_s(0)-1}{b_1 K_p}$$

dove $Z_s(0)=R_0+R$. Per errori di questo tipo, l'errore di stima dipende dal valore di K_p. Se $K_p=0$, cioè l'algoritmo equivale al solo Coulomb Counting, si ha che l'errore tende a $-\infty$, questo perché si ha una componente costante all'ingresso dell'integratore che

fa diverge l'uscita; se invece $K_p \to +\infty$, l'errore tende alla quantità costante $I_{err} Z_S(0)/b_1$ e quindi esiste un valore di K_p che annulla l'errore di stima del SOC in presenza di un errore sulla corrente:

$$K_p Z_s(0) - 1 = 0 \;\Rightarrow\; K_{p_{opt}} = \frac{1}{Z_s(0)} = \frac{1}{R_0 + R}$$

Si noti che il valore di $K_{p_{opt}}$ non è costante visto che i valori di R_0 ed R vengono aggiornati periodicamente dall'algoritmo d'identificazione online dei parametri.

3.3.3 Comportamento in presenza di un'inizializzazione errata del SOC

Uno dei problemi del Coulomb Counting è quello dell'errore causato da un'inizializzazione non corretta del valore del SOC. Il valore del SOC iniziale, SOC_{init}, non è semplice da calcolare e l'unico modo per conoscerlo accuratamente è quello di essere in uno stato noto (tipicamente quello di cella completamente carica). Per analizzare il comportamento in presenza di un errore di inizializzazione in ingresso al modulo d'integrazione, $SOC_{init_{err}}$, si ricava l'uscita del sistema facendo riferimento a *Figura 8:*

$$SOC(s) = SOC_{init_{err}} - \frac{b_1 K_p SOC(s)}{s Q_R} \;\Rightarrow\; \frac{SOC(s)}{SOC_{init_{err}}} = \frac{s}{s + \frac{b_1 K_p}{Q_R}}$$

da cui, applicando il teorema del valore finale, si ottiene il seguente errore di stima a regime:

$$\lim_{t\to\infty} SOC_{err}(t) = \lim_{s\to 0} s\,SOC_{err}(s) = \lim_{s\to 0} s\frac{SOC_{init_{err}}}{s}\frac{s}{s+\frac{b_1 K_p}{Q_R}} = 0$$

Si nota che l'algoritmo proposto è in grado di compensare completamente l'errore dovuto ad un valore non corretto d'inizializzazione dello stato di carica. In particolare, tale correzione avviene con un andamento esponenziale con costante di tempo pari a $b_1 K_p / Q_R$.

3.4 Algoritmo di stima online dei parametri

Il blocco di identificazione online dei parametri utilizza gli ingressi misurati per calcolare i parametri del modello utilizzato nell'algoritmo di stima del SOC. La struttura dell'algoritmo presentato è riportata in Figura 9.

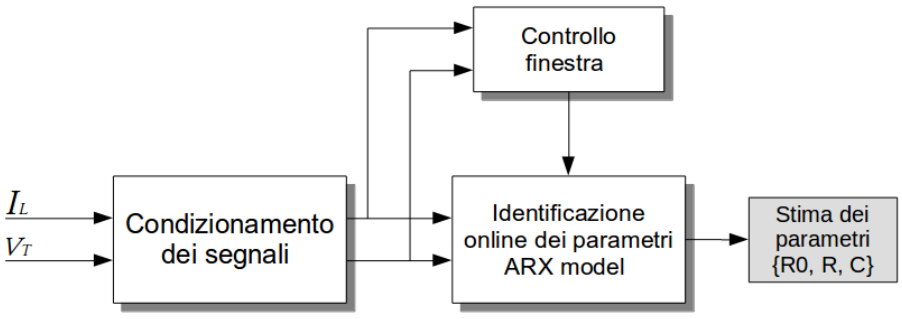

Figura 9: Schema a blocchi dell'algoritmo d'identificazione online dei parametri.

La parte relativa al condizionamento dei segnali trasforma i segnali in ingresso in modo che abbiano delle particolari caratteristiche necessarie perché lo stimatore effettui una

corretta identificazione. Il blocco di controllo della finestra verifica che queste caratteristiche siano rispettate e di conseguenza abilita il blocco ARX model che effettua una nuova stima dei parametri utilizzando il metodo del Moving Window Least Square.

3.4.1 Caratterizzazione del sistema

L'algoritmo d'identificazione dei parametri utilizza un certo numero di campioni delle grandezze i_L e v_T, presi in una finestra temporale limitata, per eseguire la stima dei parametri secondo il metodo del MWLS [4][26].

Nel caso in questione siamo in presenza di un sistema lineare tempo-invariante (LTI) i cui ingressi provengono da misurazioni ordinate nel tempo e quindi la relazione di ingresso/uscita può essere espressa tramite un modello ARX (Auto Regressive eXogenous Model), come descritto nel capitolo precedente. In particolare, viene utilizzato il modello con errore di equazione descritto dalla seguente relazione:

$$A(q)y(q)=C(q)u(q)+e(q)$$

dove

$$A(q)=1+a_1 q^{-1}+...+a_n q^{-n}$$

$$C(q)=c_0+c_1 q^{-1}+...+c_m q^{-m}$$

ed il termine *e(q)* rappresenta un rumore gaussiano bianco a media nulla. Nella notazione utilizzata *q* è l'operatore di ritardo unitario e i coefficienti *c* vanno a sostituire i coefficienti *b*, utilizzati nella notazione classica del modello ARX.

Grazie a questo modello è possibile stimare le incognite *A* e *C* (parametri da identificare), ad un certo istante, partendo dal valore dei campioni delle uscite e degli ingressi agli istanti precedenti. Queste incognite possono essere calcolate minimizzando ad ogni istante lo scarto tra le uscite stimate dal modello ARX e quelle osservate sperimentalmente, utilizzando il metodo dei minimi quadrati. Visto che i campioni dei segnali vengono aggiornati durante l'esecuzione del sistema, deve essere utilizzato l'algoritmo dei minimi quadrati ricorsivi (RLS) o del MWLS. In particolare si è scelto il secondo algoritmo perché risulta meno complesso da realizzare in sistemi embedded. Questo metodo consiste nell'eseguire periodicamente l'algoritmo LS su un lotto di *L* dati acquisiti in una finestra temporale che va da *t-L* a *t*. La lunghezza *L* della finestra deve essere scelta in modo da permettere allo stimatore di rivelare la dinamica del sistema. Per poter applicare il modello ARX alla stima dei parametri del modello di cella {b_0, b_1, R_0, R, C} bisogna innanzitutto ricavare la funzione di trasferimento tra la tensione ai terminali v_T e la corrente di batteria i_L, nel dominio tempo discreto. Questa può essere ricavata considerando il modello della cella precedentemente linearizzato e partendo dal sistema di equazioni nel dominio del tempo che lo descrive:

$$\begin{cases} \frac{d}{dt} SOC(t) = -\frac{i_L(t)}{Q_r} \\ \frac{d}{dt} v_{RC}(t) = -\frac{v_{RC}(t)}{RC} + \frac{i_L(t)}{C} \\ v_T(t) = b_0 + b_1 SOC(t) - R_0 i_L(t) - v_{RC}(t) \end{cases}$$

Applicando la trasformata di Laplace si ha che:

$$\begin{cases} s\,SOC(s) = -\dfrac{I_L(s)}{Q_r} \\ s\,V_{RC}(s) = -\dfrac{V_{RC}(s)}{RC} + \dfrac{I_L(s)}{C} \\ V_T(s) = b_0 + b_1 SOC(s) - R_0 I_L(s) - V_{RC}(s) \end{cases}$$

Dalle prime due equazioni si ottiene:

$$SOC(s) = -\frac{I_L(s)}{s\,Q_R}$$

$$V_{RC}(s) = \frac{R\,I_L(s)}{1 + RCs}$$

e sostituendole nella terza equazione si ha:

$$V_T(s) = b_0 - b_1 \frac{I_L(s)}{s\,Q_R} - R_0 I_L(s) - \frac{R\,I_L(s)}{1 + RCs}$$

Indicando con Y(s) l'uscita del sistema e con U(s) il suo ingresso si può ricavare la funzione di trasferimento del sistema:

$$\frac{Y(s) - b_0}{U(s)} = \frac{R_0 s^2 + \left(\dfrac{b_1}{Q_R} + \dfrac{R_0}{RC} + \dfrac{1}{C}\right)s + \dfrac{b_1}{RC\,Q_R}}{\left(s + \dfrac{1}{RC}\right)s}$$

Per passare nel dominio discreto (dominio della variabile *z*) si applica la trasformazione di Tustin effettuando la sostituzione

$$s \rightarrow \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

per ottenere la funzione di trasferimento discreta con tempo di discretizzazione T [5]:

$$\frac{Y(z^{-1})-b_0}{U(z^{-1})}=\frac{c_0+c_1 z^{-1}+c_2 z^{-2}}{1+a_1 z^{-1}+a_2 z^{-2}}$$

dove

$$c_0=\frac{2b_0 T RC-b_1 T^2+2Q_R RT-4Q_R R_0 RC}{2Q_R T+4Q_R RC}$$

$$c_1=\frac{-b_1 T^2+4Q_R R_0 RC}{Q_R T+2Q_R RC}$$

$$c_2=\frac{-2b_1 T RC-T^2 b_1-2Q_R RT-2Q_R R_0 T-4Q_R R_0 RC}{2Q_R T+4Q_R RC}$$

$$a_1=\frac{-4RC}{2RC+T}$$

$$a_2=\frac{2RC-T}{2RC+T}$$

In accordo con la funzione di trasferimento discreta, la relazione tra i campioni d'ingresso e quelli di uscita, nel dominio del tempo, si può esprimere come segue:

$$y(k)=-a_1 y(k-1)-a_2 y(k-2)+b_0(1+a_1+a_2)+c_0 u(k)+c_1 u(k-1)+c_2 u(k-2)$$

dove $1+a_1+a_2=0$ (relazione imposta dalle equazioni di a_1 e a_2). Ciò comporta che il termine b_0 non ha effetto sulla stima di $y(k)$ e degli altri parametri e quindi ha la funzione di un offset sull'uscita che non influenza il comportamento dinamico del sistema.

A questo punto, risolvendo il sistema di equazioni dei coefficienti della funzione di trasferimento {c_0, c_1, c_2, a_1, a_2} si ottiene un'espressione univoca dei parametri della batteria {b_0, b_1, R_0, R, C}. In particolare, dalla quarta equazione si ottiene:

$$R=-\frac{T}{2C}\frac{a_1}{2+a_1}$$

e sostituendo il valore di R ottenuto nelle prime tre equazioni si ricava il seguente sistema:

$$\begin{cases} X1=T^2\left(-1+\frac{a_1}{2+a_1}\right)b_1+2Q_RT\left(-1+\frac{a_1}{2+a_1}\right)R_0+Q_RT^2\left(\frac{a_1}{2+a_1}\right)\frac{1}{C} \\ X2=-2T^2b_1-4Q_RT\left(\frac{a_1}{2+a_1}\right)R_0 \\ X3=-T^2\left(1+\frac{a_1}{2+a_1}\right)b_1+2Q_RT\left(1+\frac{a_1}{2+a_1}\right)R_0-Q_RT^2\left(\frac{a_1}{2+a_1}\right)\frac{1}{C} \end{cases}$$

dove

$$X1=2Q_RT\left(1-\frac{a_1}{2+a_1}\right)c_0$$

$$X2=2Q_RT\left(1-\frac{a_1}{2+a_1}\right)c_1$$

$$X3=2Q_RT\left(1-\frac{a_1}{2+a_1}\right)c_2$$

3.4.2 Descrizione e funzionamento dell'algoritmo

Per poter identificare i coefficienti {c_0, c_1, c_2, a_1, a_2} viene utilizzato il Moving Window Least Square, algoritmo eseguito dal blocco d'identificazione online dei parametri. La stima in ciascun istante di tempo viene effettuata su un insieme di campioni degli ingressi e delle uscite precedentemente acquisiti in una finestra temporale (window). I passi da eseguire sono i seguenti:

1. la stima all'istante k, si effettua inviando allo stimatore L campioni della tensione e della corrente corrispondenti agli istanti [$(k-L+1)$, $(k-L+2)$, …, k], scegliendo opportunamente la lunghezza della finestra L in modo che la stima sia corretta cioè in modo tale da permettere allo stimatore di osservare le dinamiche di interesse della batteria;
2. viene eseguita l'identificazione dei parametri con il metodo LS;
3. si sposta la finestra temporale di un passo stabilito N e si aggiorna la finestra con i nuovi campioni riferiti agli istanti [$(k+N-L+1)$, $(k+N-L+2)$, …, $k+N$];
4. si inviano i nuovi campioni allo stimatore e si riparte dal punto 2.

Il modello ARX ha come ingresso la corrente che scorre nella batteria (detta segnale di stimolo) e come uscita la tensione ai suoi capi (detta segnale di risposta). Perché la stima sia corretta il segnale d'ingresso deve rispettare la condizione di persistente eccitazione. Infatti, il segnale di stimolo non deve essere costante all'interno della finestra temporale e deve sempre variare abbastanza da eccitare tutte le dinamiche del sistema.

Solitamente, la lunghezza della finestra viene scelta sulla base di una conoscenza generale del sistema, ma questo non basta a prevenire determinate circostanze in cui l'ingresso non rispetta la persistente eccitazione. Per questo motivo l'algoritmo è

provvisto di un controllo che permette di prevenire delle identificazioni errate (controllo finestra). Per assicurarci che nella finestra di osservazione il sistema sia stato eccitato adeguatamente, questo blocco di controllo calcola il valore della deviazione standard della corrente, a partire dai campioni contenuti nella finestra stessa, che viene poi confrontato con una soglia impostata dall'utente. Se la deviazione standard è minore della soglia stabilita l'identificazione non viene effettuata e il modello conserva i valori dei parametri calcolati durante l'identificazione precedente.

Il blocco di condizionamento dei segnali si occupa invece di evitare problemi dovuti al rumore causato dal processo di misura. Infatti, un eccessivo rumore fa si che i dati non rappresentino la dinamica dominante del sistema. I segnali in ingresso vengono quindi filtrati tramite un filtro passa-basso e poi decimati. In questo modo i segnali inviati al blocco di stima vengono campionati con un periodo maggiore e le componenti eccessivamente rumorose vengono scartate. Nell'eseguire queste due operazioni bisogna rispettare la condizione di Nyquist.

All'ingresso dello stimatore vengono forniti, ad un certo istante *t*, due vettori di campioni, uno relativo alla corrente d'ingresso, $\underline{u}$, e l'altro alla tensione d'uscita, $\underline{y}$, contenuti nella finestra di lunghezza *L*:

$$\underline{y}=[y(t-L+1),...,y(t)]^T$$

$$\underline{u}=[u(t-L+1),...,u(t)]^T$$

La relazione tra i campioni di ingresso e quelli di uscita può essere rappresentata nella seguente forma compatta:

$$y(t)=-a_1 y(t-1)-a_2 y(t-2)+c_0 u(t)+c_1 u(t-1)+c_2 u(t-2) \;\Rightarrow\; y(t)=\varphi(t)\underline{\theta}$$

dove

$$\varphi(t)=[-y(t-1) \quad -y(t-2) \quad u(t) \quad u(t-1) \quad u(t-2)]$$

$$\underline{\theta}=[a_1 \quad a_2 \quad c_0 \quad c_1 \quad c_2]^T$$

e, con gli *L* campioni acquisiti, si può realizzare una matrice *Φ* di dimensione *L x 5* le cui righe sono costituite da $\varphi(t)$:

$$\varphi(t-L+1)=[0 \quad 0 \quad u(t-L+1) \quad 0 \quad 0]$$

$$\varphi(t-L+2)=[-y(t-L+1) \quad 0 \quad u(t-L+2) \quad u(t-L+1) \quad 0]$$

$$\vdots$$

$$\varphi(t)=[-y(t-1) \quad -y(t-2) \quad u(t) \quad u(t-1) \quad u(t-2)]$$

Quindi il sistema può essere scritto come $\underline{y}=\Phi\underline{\theta}$, dove:

$$\Phi=\begin{bmatrix}\varphi(t-L+1)\\ \varphi(t-L+2)\\ \vdots\\ \varphi(t)\end{bmatrix}$$

A questo punto si applica al sistema una delle tecniche di risoluzione dei problemi LS per determinare un'approssimazione di $\underline{\theta}$ nel senso dei minimi quadrati. Nell'algoritmo

presentato in questo elaborato si utilizza la fattorizzazione QR con Givens rotation. Si è preferito usare la rotazione di Givens rispetto all'algoritmo CORDIC in quanto il sistema lavora con numeri in virgola mobile (single precision), e il CORDIC porta seri vantaggi solo nel caso in cui si lavora in virgola fissa.

3.4.3 Fattorizzazione QR con Givens rotation

La QRD con Givens rotation permette di calcolare la fattorizzazione QR di una matrice *A* tramite un metodo iterativo che usa una matrice elementare di Givens, detta matrice di rotazione:

$$G(i,j,\alpha)=\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

Questa matrice è ortonormale e viene costruita nel seguente modo, per $i>j$:

- si impone che $g_{ii}=g_{jj}=c=\cos(\alpha)$;
- tutti gli elementi g_{kk} sulla diagonale si mettono pari a 1, fatta eccezione per gli elementi g_{ii} e g_{jj} ;
- si impone $g_{ij}=s=\sin(\alpha)$ e $g_{ji}=-s=-\sin(\alpha)$;
- tutti gli altri elementi sono pari a 0.

Moltiplicando $G^T(i,j,\alpha)$ per un vettore $x\in\mathbb{R}^m$, in modo da ottenere $y=G^T(i,j,\alpha)x$, si ha che solo i termini x_i e x_j saranno interessati dal calcolo:

$$y_k = \begin{cases} c\,x_i - s\,x_j & k = i \\ s\,x_i + c\,x_j & k = j \\ x_k & k \neq i, j \end{cases}$$

In particolare, tale matrice opera una rotazione del vettore sul piano identificato dai versori e_i ed e_j. Si nota che è possibile ottenere $y_j = 0$ imponendo che:

$$c = \cos(\alpha) = \frac{x_i}{\sqrt{x_i^2 + x_j^2}} \qquad , \qquad s = \sin(\alpha) = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}$$

Sotto queste condizioni, moltiplicando $G^T(i, j, \alpha)$ per *A* si ottiene una matrice il cui elemento *(i,j)*-esimo è nullo. Eseguendo opportunamente quest'operazione in modo ricorsivo è possibile annullare tutti gli elementi di ogni colonna di *A* sotto la diagonale principale, ottenendo così una matrice triangolare. In particolare, si sceglie come x_i un elemento facente parte della diagonale e come x_j un elemento appartenente alla stessa colonna, ma situato sotto la diagonale. Alla fine dell'algoritmo si ottiene una matrice $R \in \mathbb{R}^{m \times n}$, la cui sotto-matrice più in alto è triangolare superiore, ed una matrice ortonormale $Q \in \mathbb{R}^{m \times m}$, ottenuta dal prodotto delle *G*:

$$R = G^T(m,1)\,G^T(m-1,1)\cdots G^T(m,n)\,A = \prod_{j=1}^{n} \prod_{i=1}^{m-j} G^T(m-i+1, j)\,A$$

$$Q = G(m,1)\,G(m-1,1)\cdots G(m,n)¿ = \prod_{j=1}^{n} \prod_{i=1}^{m-j} G(m-i+1, j)$$

L'algoritmo complessivo può essere descritto dal seguente codice “Matlab-like”:

$Q = I_m$
for $j = 1:n$
  for $i = m:-1:(j+1)$
    $[c,s] = givens(A(i-1,j), A(i,j))$

$$A(i-1:i, j:n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T A(i-1:i, j:n)$$

$$Q(i-1:i, j:n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} Q(i-1:i, j:n)$$

  end
end

La funzione *givens(a,b)* serve a calcolare i valori di *c* e *s*, tali che

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

e per farlo utilizza il seguente algoritmo:

function $[c,s] = givens(a,b)$
  if $b = 0$
    $c = 1; s = 0$
  else
    if $|b| > |a|$
      $\tau = -a/b; s = 1/\sqrt{1+\tau^2}; c = s\tau$
    else
      $\tau = -b/a; c = 1/\sqrt{1+\tau^2}; s = c\tau$
    end
  end

Come descritto nel capitolo precedente, successivamente alla decomposizione QR, bisogna eseguire delle sostituzioni per ottenere il valore dei coefficienti {c_0, c_1, c_2, a_1, a_2}. Una volta ricavati i valori di questi coefficienti bisogna risolvere il sistema di equazioni

univoche ottenuto nel paragrafo 3.4.1 per ottenere i valori dei parametri del circuito equivalente della batteria.

4 Implementazione dell'algoritmo

In questo capitolo vengono descritti gli aspetti implementativi per la realizzazione dell'algoritmo su FPGA. In particolare, vengono presentati il flusso di progetto e i CAD utilizzati e successivamente vengono analizzate nel dettaglio tutte le parti dell'algoritmo.

4.1 Flusso di progetto

Gli FPGA (Field Programmable Gate Array) sono sempre più utilizzati in applicazioni di tipo digital signal proccessing. Essi non riescono a raggiungere le prestazioni e le frequenze di clock dei processori general-purpose o dei DSPs (Digital Signal Processors), ma comunque portano notevoli vantaggi nel caso in cui è possibile realizzare una parallelizzazione del flusso di dati. Il problema principale si ha nei sistemi in virgola mobile (single o double precision) in quanto la loro realizzazione risulta difficoltosa a causa di latenze nel processing e di congestione del routing, oltre al fatto che la descrizione hardware RTL (Register-Transfer-Level) in Verilog o VHDL non si adatta bene all'utilizzo di dati in virgola mobile. Questi problemi portano a dei cattivi risultati in termini di rapporto prestazioni/costi [27].

Nel sistema in questione la parallelizzazione del flusso di dati può portare a molti vantaggi in quanto l'algoritmo può essere implementato come acceleratore hardware che lavora in parallelo ad un microcontrollore e che si occupa di calcolare il SOC di tutte le celle della batteria. Il sistema può essere sviluppato anche come un SoPC (System on a Programmable Chip), implementando la stima del SOC e l'identificazione dei

parametri come periferiche o come coprocessori hardware del processore principale.

Il sistema in questione viene implementato utilizzando la rappresentazione in virgola mobile, single precision. L'utilizzo della virgola mobile rende molto più rapida la traduzione in hardware dell'algoritmo poiché non bisogna affrontare tutte le problematiche derivanti dalla rappresentazione in virgola fissa. Infatti, questa rappresentazione numerica, a differenza della virgola fissa, consente di ottenere un errore costante per un range di valori molto ampio pur mantenendo invariato il numero di bit necessari alla rappresentazione.

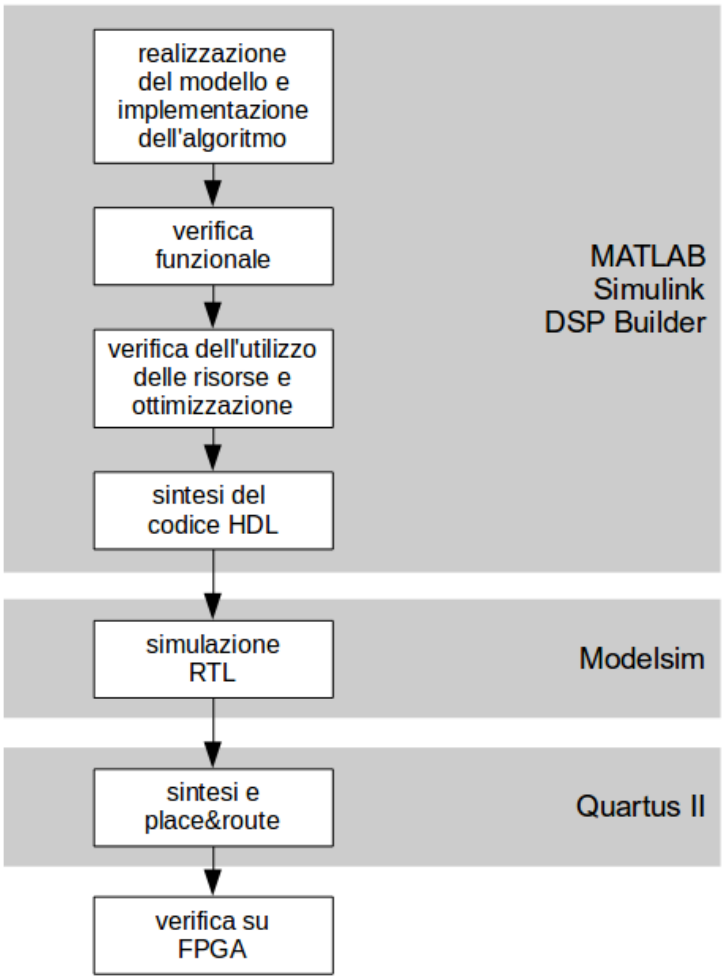

Figura 10: Flusso di progetto.

Per ovviare alle problematiche di sviluppo su FPGA per sistemi di questo tipo si è seguito un flusso di progetto innovativo (Figura 10) che prevede l'utilizzo di un tool di sviluppo Altera, DSP Builder, che permette di ridurre i tempi e la complessità di progettazione. L'algoritmo viene implementato realizzando un modello con DSP Builder. Questo tool opera su Simulink, in ambiente MATLAB, aggiungendovi delle librerie specifiche (block-sets) che permettono di realizzare sistemi di digital signal processing in maniera semplice e veloce [27]. In particolare, permette di realizzare il modello di un sistema e di verificarne il corretto funzionamento con gli strumenti tipici di Simulink e MATLAB.

Successivamente, impostando la frequenza di clock, la frequenza di campionamento degli ingressi e definendo il dispositivo su cui si vuole realizzare il sistema (FPGA Altera), il DSP Builder realizza una descrizione hardware, espressa in codice VHDL, a partire dall'algoritmo sviluppato e simulato ad alto livello in Simulink. Questo permette di creare il sistema senza conoscere nel dettaglio il dispositivo e di generare un'implementazione che può essere utilizzata su diverse famiglie di FPGA Altera con architetture hardware differenti. Il DSP Builder, nel generare la descrizione hardware, presenta i seguenti vantaggi [27][28]:

- pipelining automatico che permette di raggiungere la frequenza desiderata (fino a 300-400 MHz), risolvendo in automatico i time constraints;
- condivisione automatica delle risorse;
- realizzazione di sistemi in virgola mobile ad alte prestazioni, con l'ausilio del metodo “fused datapath”. In genere, le operazioni in virgola mobile sono caratterizzate da strutture complesse e molti stadi di pipeline che causano

un'elevata latenza ed una congestione del routing. Il DSP Builder risolve questo problema unendo le varie parti del datapath, combinando i singoli operatori fondamentali in un'unica funzione o datapath. In questo modo viene eliminata la ridondanza tipica che viene presentata dai classici sistemi in virgola mobile, ottimizzando le risorse utilizzate e riducendo la latenza.

Il DSP Builder permette anche di avere una stima dell'utilizzo delle risorse in base al dispositivo selezionato. Tale stima non avviene a livello hardware, ma consente comunque di effettuare un'eventuale ottimizzazione, agendo su alcuni parametri offerti dal tool di sviluppo.

Insieme al codice VHDL, il DSP Builder genera anche i testbench per la verifica funzionale e timing del sistema su Modelsim. Successivamente si può includere e compilare il codice HDL in Quartus II.

Terminata la compilazione è possibile programmare il dispositivo e verificare il funzionamento del sistema su FPGA.

4.2 Descrizione generale del sistema

L'algoritmo da implementare in hardware deve riuscire a calcolare lo stato della batteria in modo autonomo ed in parallelo alle operazioni svolte dall'unità principale. In questo modo il processore può svolgere le altre funzioni del BMS e conoscere lo stato della batteria semplicemente interrogando il modulo hardware implementato.

Un vantaggio derivante da questa implementazione è che il modulo hardware può essere utilizzato in time sharing per tutte le celle della batteria. Questo è possibile grazie al fatto che, solitamente, il SOC varia molto lentamente e dunque la stima può

essere effettuata con una frequenza (al massimo qualche Hz) molto inferiore a quella di clock del sistema (dell'ordine delle centinaia di MHz).

L'algoritmo può essere suddiviso in più blocchi che comunicano tra loro. Ad esempio si potrebbe avere un blocco che calcola il SOC, uno per il SOH ed uno che effettua l'identificazione online dei parametri. In questo lavoro di tesi è stato realizzato il blocco di stima dello stato di carica e quello dell'identificazione online dei parametri, che verranno analizzati nel dettaglio nei prossimi paragrafi.

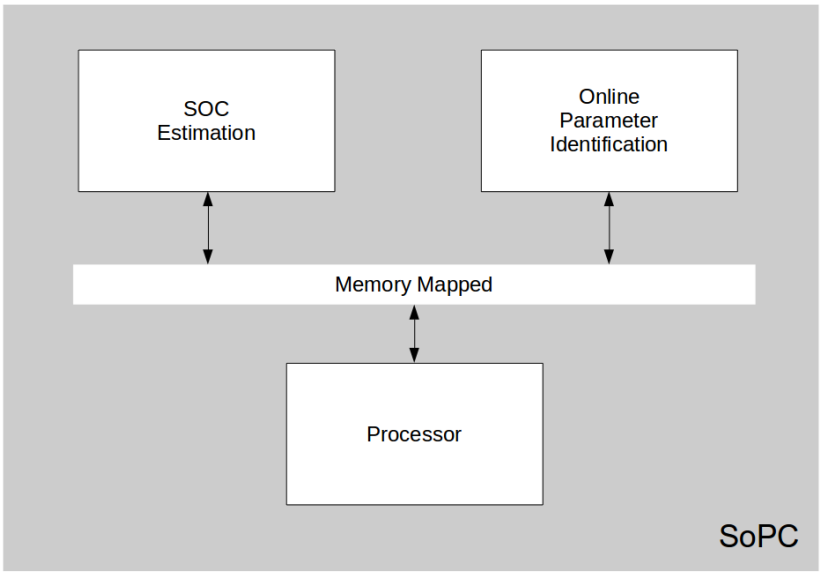

Figura 11: Architettura FPGA del sistema.

Questi blocchi sono stati sviluppati come periferiche custom che comunicano tra loro grazie ad un modulo hardware (Memory Mapped) che memorizza le uscite dei due blocchi in un set di registri accessibili dal processore all'interno del SoPC; inoltre, genera e fornisce i segnali di abilitazione e gli ingressi necessari per la corretta esecuzione dell'algoritmo (Figura 11).

4.3 Stima del SOC

Il blocco di stima del SOC è rappresentato in Figura 12, dove è riportata la top level entity del modulo realizzato e in cui sono visibili gli ingressi e le uscite dello stesso. I segnali di valid e channel (*v_i* e *c_i*) permettono di segnalare quando i campioni in ingresso sono validi e a quale cella appartengono. In ingresso devono essere forniti: i campioni delle grandezze misurate della batteria (*I_L* e *V_t*), il valore dei parametri del modello derivanti dal blocco di identificazione (*R*, *R0* e *C*), il periodo di campionamento *T*, i valori della tensione sul gruppo RC (*Vrc_rit*), dello stato di carica (*SoC_in*) e della tensione calcolata dal modello (*Vm_in*), relativi al ciclo precedente e il valore della carica massima Q_R della batteria, contenuto nella costante *E1*.

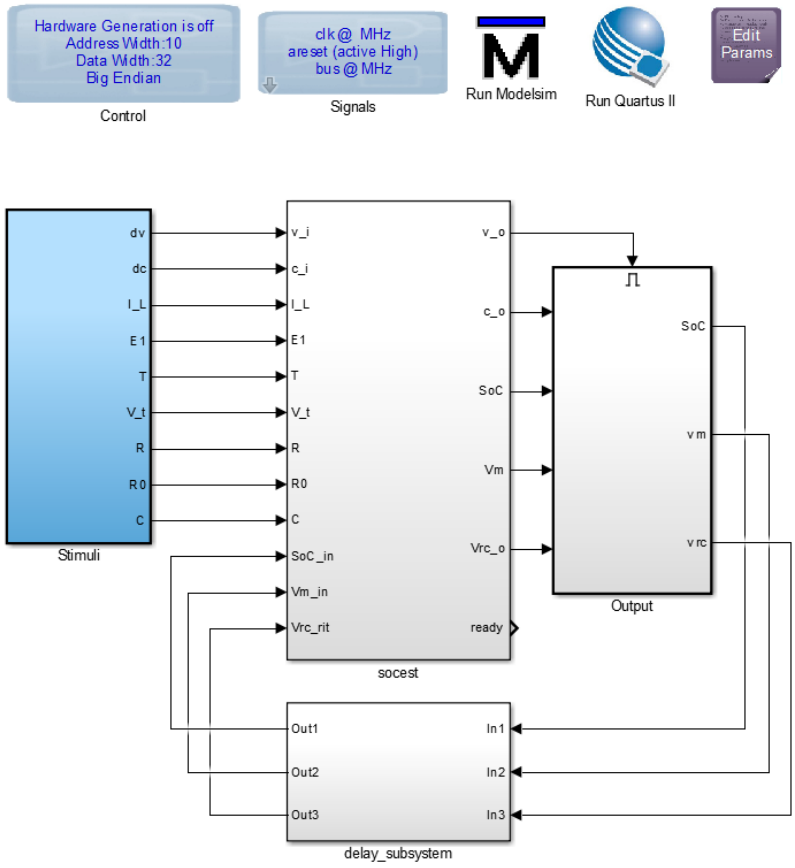

Figura 12: Top level entity del modulo di stima del SOC.

Tutti i valori in ingresso al blocco vengono memorizzati in un set di registri, uno per ogni cella. In questo modo, ogni volta che si effettua il calcolo per una determinata cella, basta scambiare il registro dal quale si prelevano i dati. In uscita invece si hanno i segnali di valid e di channel (*v_o* e *c_o*) per identificare quando sono validi i risultati del calcolo. Il segnale *ready* serve a stabilire quando è possibile fornire dei nuovi ingressi al modulo. Si trovano poi i segnali di *SoC*, *Vm* e *Vrc_o* calcolati dal modulo e relativi al ciclo attuale. Il livello gerarchico inferiore del modulo è rappresentato in Figura 13, dove si possono facilmente individuare tutti i blocchi, già presentati in Figura 6, che occorrono al calcolo dello stato di carica.

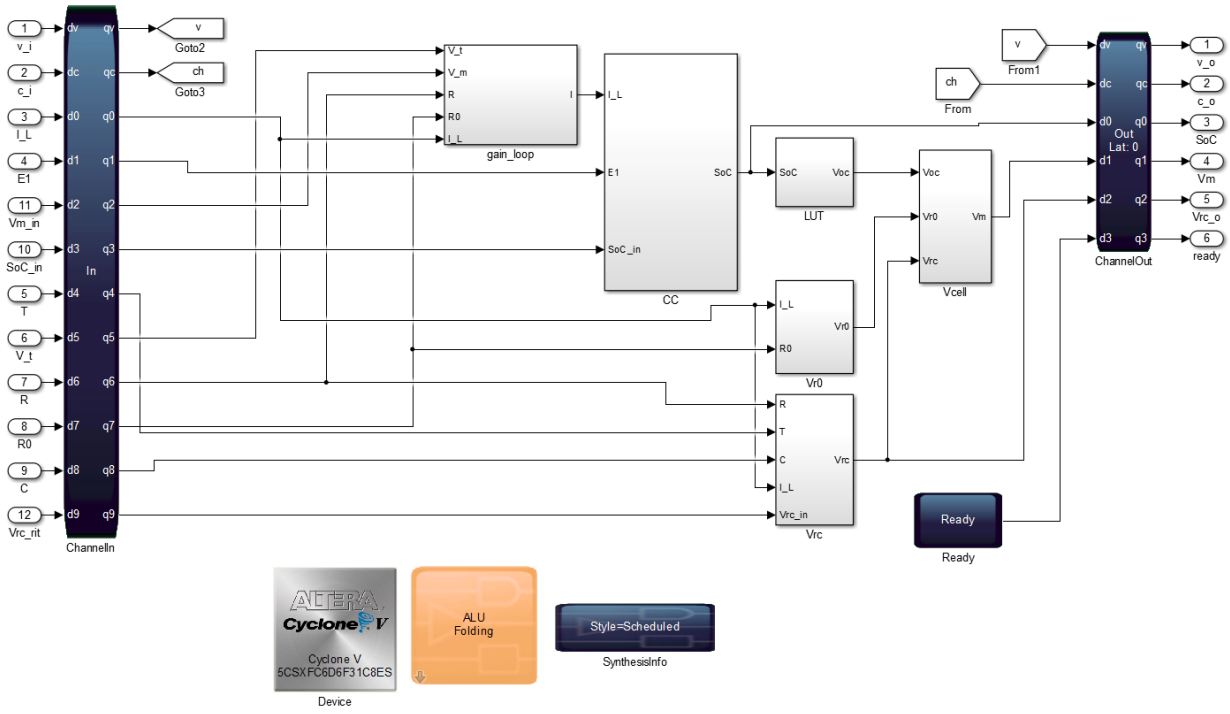

Figura 13: Modulo di stima del SOC.

Il blocco CC si occupa di eseguire il Coulomb Counting (Figura 14), cioè svolge un'operazione di integrazione per calcolare il valore del SOC a partire dal suo valore

iniziale, fornito all'ingresso *SoC_in*. Il calcolo effettuato è quindi il seguente:

$$SOC(t) = SOC_{init} - \frac{1}{Q_R}\int_{t_0}^{t} i_L(\tau)\,d\tau$$

e, applicandovi la trasformata di Laplace, si ottiene:

$$SOC(s) = -\frac{I_L(s)}{sQ_R}$$

Grazie alla trasformazione di Tustin si può passare al dominio discreto:

$$SOC(z) = -\frac{I_L(z)}{Q_R}\frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}$$

da cui si ottiene che:

$$SOC(z) = SOC(z)z^{-1} - \frac{T}{2Q_R}I_L(z)(1+z^{-1})$$

La moltiplicazione di un segnale per z^{-1} nel dominio discreto equivale a ritardare lo stesso di un periodo di campionamento. Dunque il SOC all'istante attuale dipende da quello all'istante precedente, dalla corrente attuale e precedente (la cui somma costituisce un passo d'integrazione). Il termine $T/2Q_R$ viene fornito dall'esterno con la costante *E1*. Il valore del SOC calcolato viene successivamente limitato nell'intervallo di valori da 0 a 100 da un insieme di elementi logici. Questo avviene perché in questa implementazione il SOC è calcolato in percentuale.

La corrente all'istante attuale (*I_L*) viene memorizzata in una memoria (blocco presente tra le librerie del DSP Builder) che aggiorna i propri dati ogni volta che viene abilitato il modulo, usando come indirizzo il valore del canale in ingresso (Figura 15). Il valore memorizzato viene poi usato nel ciclo successivo e fornito all'ingresso *I_L_rit*. L'utilizzo di questa metodologia permettere di effettuare il time sharing tra le celle.

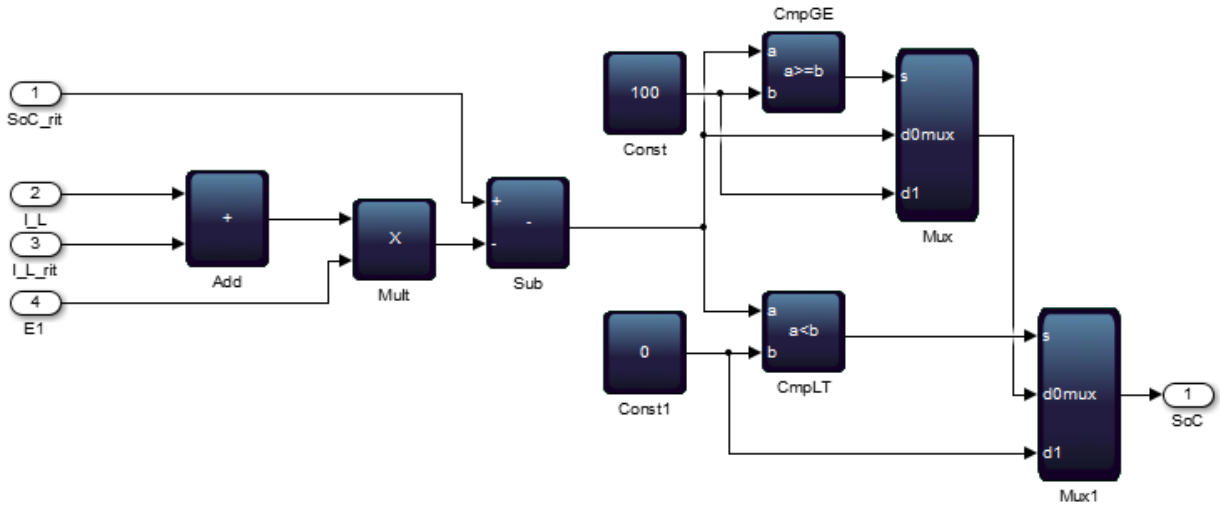

Figura 14: Modulo che esegue il Coulomb Counting.

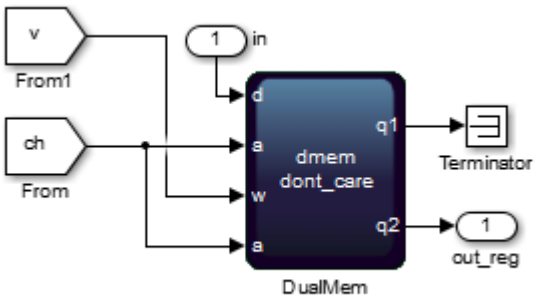

Figura 15: Blocco di memoria del DSP Builder usato per memorizzare i valori dei campioni da ritardare.

Il blocco Vrc calcola il valore della tensione ai capi del parallelo tra R e C. Il valore di

questa tensione è dato da:

$$\frac{d}{dt}v_{RC}(t)=-\frac{v_{RC}(t)}{RC}+\frac{i_L(t)}{C}$$

e, applicando la trasformata di Laplace, si ottiene:

$$sV_{RC}(s)=-\frac{V_{RC}(s)}{RC}+\frac{I_L(s)}{C}$$

Passando poi al dominio discreto si ha che:

$$V_{RC}(z)\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}=-\frac{V_{RC}(z)}{RC}+\frac{I_L(z)}{C} \Rightarrow V_{RC}(z)=-\frac{T}{2RC}\frac{1+z^{-1}}{1-z^{-1}}V_{RC}(z)+\frac{T}{2C}\frac{1+z^{-1}}{1-z^{-1}}I_L(z) \Rightarrow$$

$$\Rightarrow V_{RC}(z)\left(1+\frac{T}{2RC}\frac{1+z^{-1}}{1-z^{-1}}\right)=\frac{T}{2C}\frac{1+z^{-1}}{1-z^{-1}}I_L(z) \Rightarrow$$

$$\Rightarrow V_{RC}(z)-V_{RC}(z)z^{-1}+\frac{T}{2RC}V_{RC}(z)+\frac{T}{2RC}V_{RC}(z)z^{-1}=\frac{T}{2C}(1+z^{-1})I_L(z) \Rightarrow$$

$$\Rightarrow V_{RC}(z)\left(\frac{2RC+T}{2RC}\right)=-V_{RC}(z)\left(\frac{T-2RC}{2RC}\right)z^{-1}+\frac{T}{2C}(1+z^{-1})I_L(z) \Rightarrow$$

$$\Rightarrow V_{RC}(z)=\frac{2RC-T}{2RC+T}V_{RC}(z)z^{-1}+\frac{RT}{2RC+T}(1+z^{-1})I_L(z) \Rightarrow$$

$$\Rightarrow V_{RC}(z)=\frac{1}{2RC+T}[(2RC-T)V_{RC}(z)z^{-1}+RTI_L(z)(1+z^{-1})]$$

Si nota che il valore attuale di $V_{RC}(z)$ dipende dal valore attuale della corrente e dai campioni precedenti di i_L e di v_{RC}. L'implementazione di tale relazione viene riportata in Figura 16.

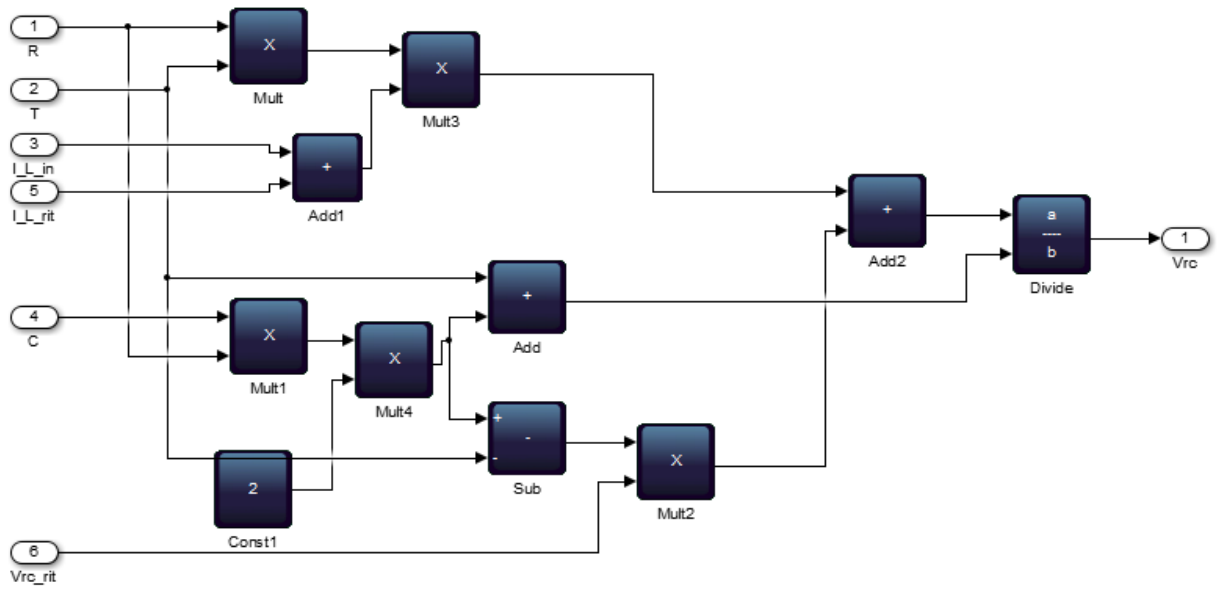

Figura 16: Modulo per il calcolo della tensione v_{RC}.

Il blocco LUT contiene una look-up table indirizzata dai valori del SOC calcolati dal blocco che esegue il Coulomb Counting. La curva OCV-SOC si suppone uguale per ogni cella in modo da evitare un consumo eccessivo di risorse (tale ipotesi trova conferma in letteratura). I valori della curva sono stati determinati con un test che verrà descritto in seguito.

Il blocco Vr0 contiene semplicemente un moltiplicatore che effettua la moltiplicazione tra i campioni di R_0 e i_L per calcolare la tensione v_{R_0} ai capi della resistenza R_0.

La tensione del modello viene calcolata dal blocco Vcell che sottrae al valore di V_{OC} i valori delle tensioni v_{RC} e v_{R_0}.

Il blocco gain_loop esegue la differenza tra i campioni di v_M e v_T, la moltiplica per il valore calcolato di $K_{p_{opt}} = 1/(R_0 + R)$ e sottrae questo risultato al valore del campione della corrente i_L.

4.4 Identificazione online dei parametri

La top level entity del modulo di stima dei parametri è riportata in Figura 17, mentre in Figura 18 è riportato il livello gerarchico inferiore.

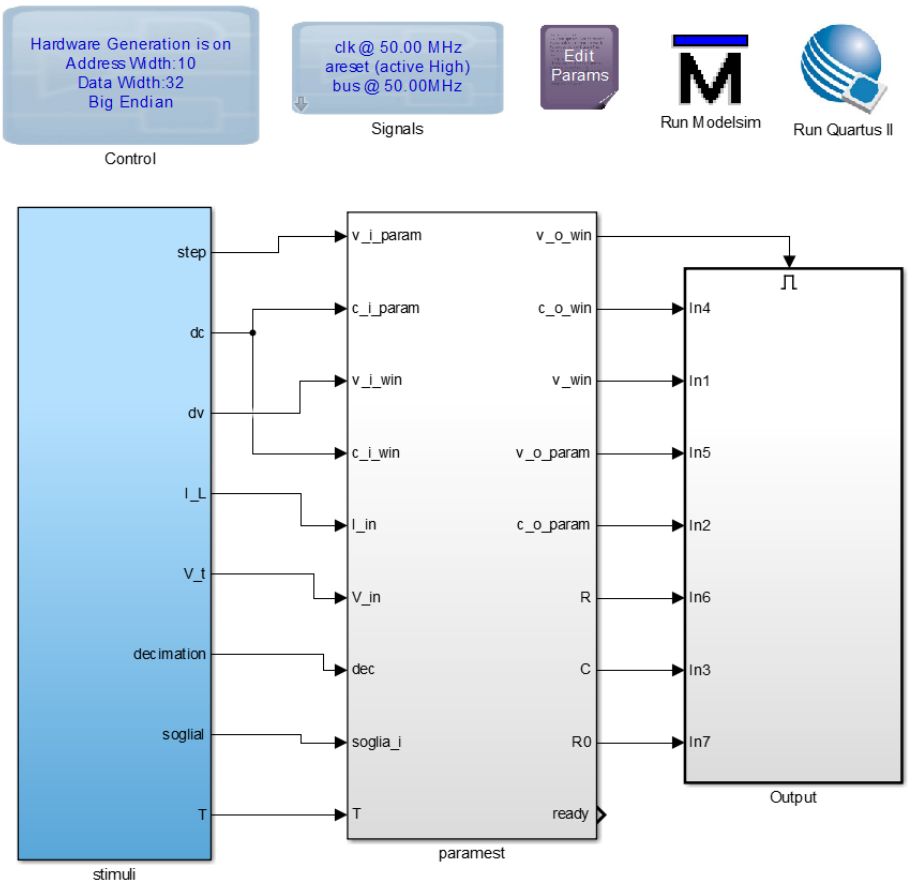

Figura 17: Top level entity del modulo di identificazione online dei parametri.

Il modulo d'identificazione si attiva ogni volta che sono disponibili dei nuovi campioni di i_L e v_T in ingresso. La validità dei campioni viene stabilità dal segnale di validazione *v_i_win*, mentre la cella a cui appartengono le misurazioni viene identificata da segnale *c_i_win*. Questi due segnali abilitano il blocco Window che ha le seguenti funzioni:

- effettua il condizionamento dei segnali, decimando i campioni in ingresso (il filtro passa-basso non è stato ancora implementato). Il segnale di decimazione (*dec*)

viene generato, insieme ai segnali d'ingresso, come specificato nel paragrafo 4.2, a partire da un fattore di decimazione definito dall'utente;

- crea i vettori *I_vect* e *V_vect* che contengono rispettivamente i campioni di corrente e di tensione relativi ad una determinata finestra temporale. Il numero di campioni dipende dalla lunghezza di tale finestra. In particolare, si ha che il numero di campioni *L* è pari a:

$$L = \frac{lunghezza\ della\ finestra\ temporale}{periodo\ di\ campionamento * fattore\ di\ decimazione}$$

- verifica che la finestra temporale sia idonea per una corretta identificazione, cioè che sia verificata la persistente eccitazione. Questa verifica viene eseguita calcolando la deviazione standard del vettore *I_vect* e confrontando il risultato con un valore di soglia fornito dall'utente all'ingresso ingressi *soglia_i*.

I due vettori in uscita da questo blocco vengono memorizzati in delle memorie contenute nel modulo Vector_Mem, ogni volta che la finestra risulta valida. Infatti, la scrittura in memoria è abilitata dal segnale *v_win*, che viene portato in uscita insieme al segnale *v_o_win* che identifica quando i dati sono validi e a *c_o_win* che indica a quale cella appartengono. Il segnale *c_o_win* viene utilizzato anche per indirizzare la memoria in fase di scrittura.

L'identificazione viene effettuata in maniera periodica, con un determinato passo di stima. Viene quindi fornito un segnale all'ingresso *v_i_param* che abilita l'algoritmo d'identificazione. Il segnale *c_i_param* viene utilizzato per indirizzare la locazione di memoria dalla quale prelevare i vettori dei campioni.

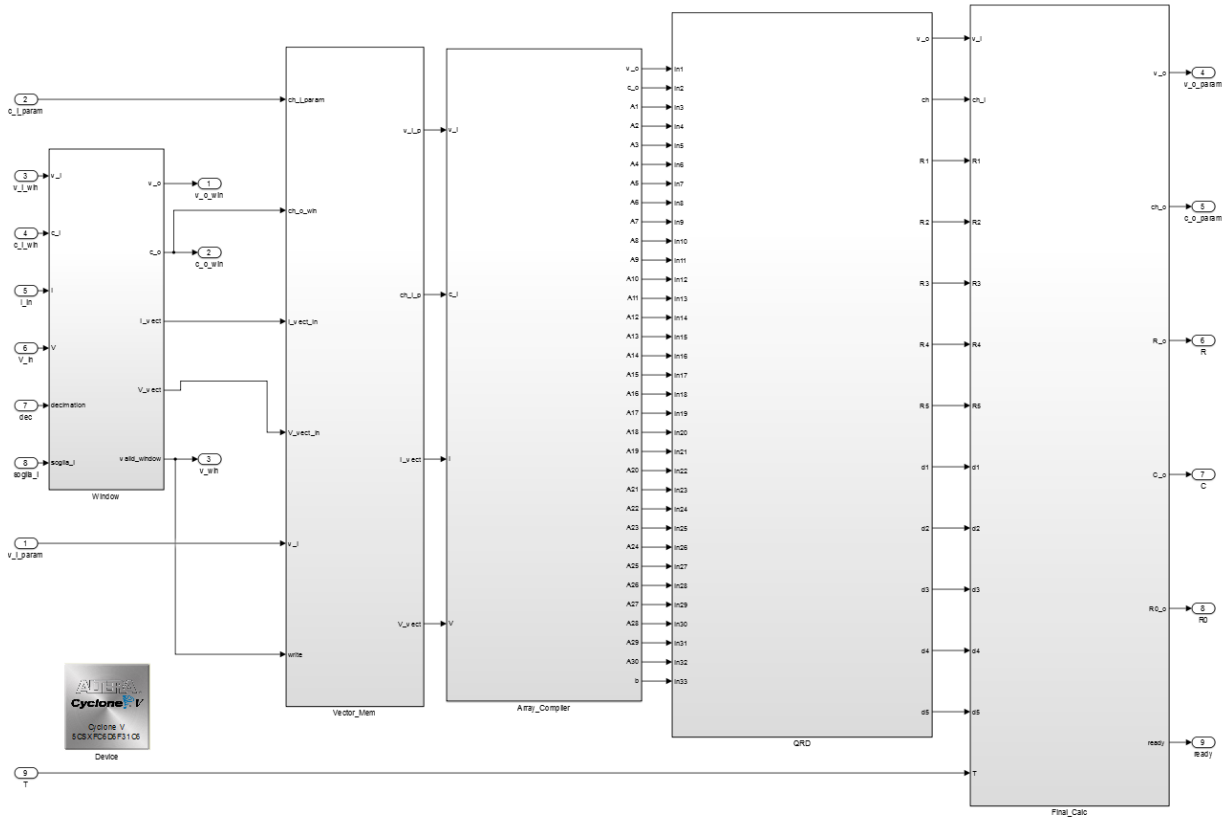

Figura 18: Modulo di identificazione online dei parametri.

A questo punto si attivano in successione gli altri tre blocchi. I due vettori vengono utilizzati dal blocco Array_Compiler per creare la matrice e il vettore dei termini noti che determinano il sistema da risolvere nel senso dei minimi quadrati ($A\,x = b$), le cui dimensioni dipendono dal numero di campioni presenti nella finestra temporale. Si attiva poi il blocco QRD che effettua la decomposizione QR e infine il blocco Final_Calc che esegue la sostituzione inversa e calcola i valori dei parametri {*R0*, *R*, *C*}.

I parametri sono validi quando si attiva il segnale *v_o_param*. Il segnale *ready* determina quando l'intero sistema è pronto a ricevere i nuovi ingressi e ad effettuare una nuova identificazione.

Di seguito viene descritta nel dettaglio l'implementazione di ogni blocco appena introdotto.

4.4.1 Window

Il modulo Window (Figura 19) acquisisce la corrente e la tensione e crea i vettori *I_vect* e *V_vect* contenenti i campioni della finestra temporale.

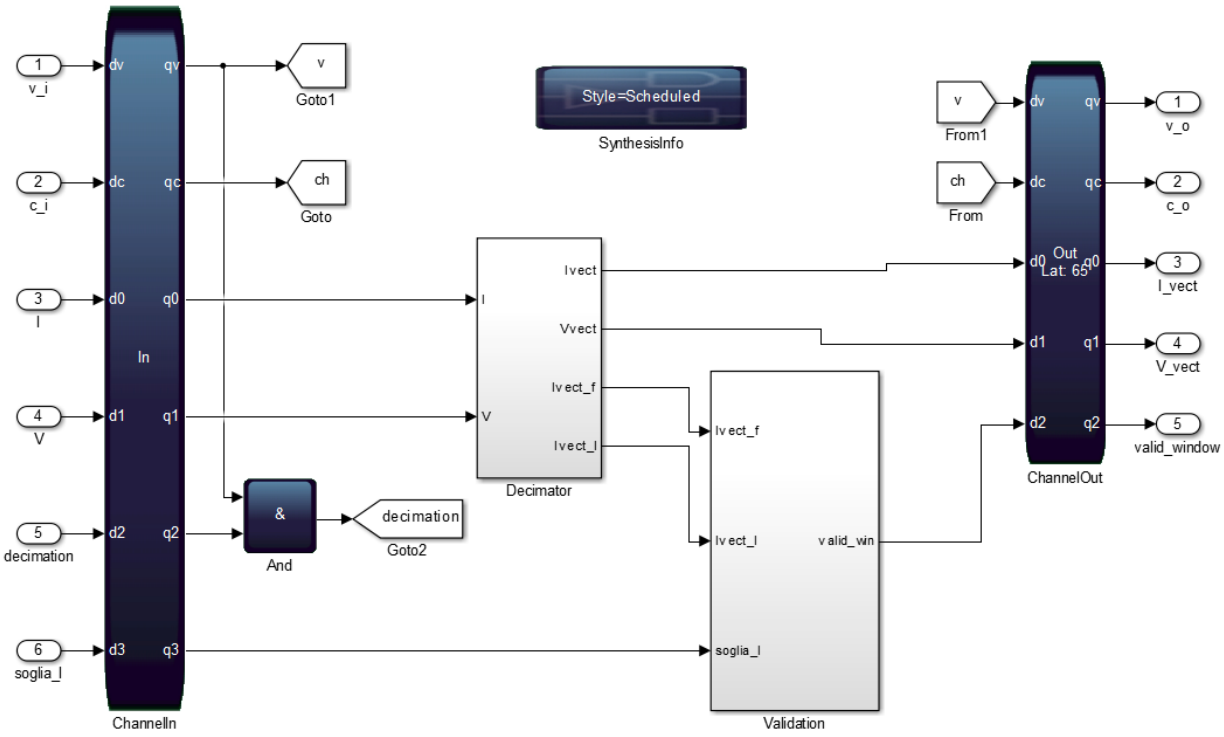

Figura 19: Modulo che effettua il condizionamento dei segnali e verifica la validità della finestra temporale.

Tali campioni vengono forniti in ingresso con un certo periodo di campionamento e vengono poi decimati grazie ad un segnale di decimazione, fornito all'ingresso *dec,* che attiva uno shift register, permettendo la memorizzazione del nuovo dato. Lo shift register è contenuto all'interno del blocco Decimator ed è realizzato con dei blocchi di memoria in modo da consentire il time sharing tra le celle. I valori in esso contenuti vengono utilizzati per realizzare i vettori d'uscita. In questi due vettori, oltre agli *L* campioni della finestra temporale relativa all'istante attuale, sono presenti anche altri due campioni relativi alla finestra precedente; il motivo verrà spiegato nei prossimi

paragrafi.

Per ogni coppia di vettori in uscita, il blocco Validation (Figura 20) genera un segnale *valid_win* che serve a stabilire la validità della finestra temporale. In particolare, la finestra temporale è valida se il vettore di corrente ha una deviazione standard maggiore o uguale alla relativa soglia.

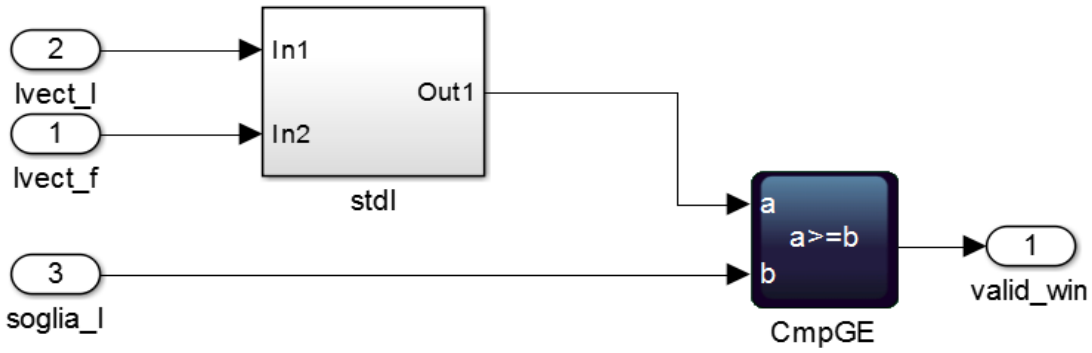

Figura 20: Modulo per la verifica della validità della finestra temporale.

In generale, la deviazione standard del vettore $\underline{x}$ di lunghezza N si può calcolare come:

$$\sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

dove x_i è l'i-esimo elemento del vettore e $\bar{x}$ è il suo valore medio, ricavabile dalla seguente relazione:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Per contenere il consumo di risorse, si può notare che:

$$(N-1)\sigma_{\underline{x}}^{2}=\sum_{i=1}^{N}(x_i-\overline{x})^2$$

e quindi fornendo come soglia il valore $(N-1)\sigma_{\underline{x}}^{2}$, si evita il calcolo di una radice quadrata e di una moltiplicazione. Inoltre, la sommatoria può essere scritta come:

$$\sum_{i=1}^{N}(x_i-\overline{x})^2=\sum_{i=1}^{N}x_i^{2}-\frac{1}{N}\left(\sum_{i=1}^{N}x_i\right)$$

e quindi può essere calcolata in maniera sequenziale (in quanto non occorre conoscere a priori il valore medio del vettore), dal modulo di calcolo della deviazione standard, visibile in Figura 20, fornendogli in ingresso il valore in ingresso e quello in uscita dallo shift register.

In definitiva, la stima verrà eseguita solo quando è soddisfatta la seguente condizione:

$$\sum_{i=1}^{L}I\_vect_i^{2}-\frac{1}{L}\left(\sum_{i=1}^{L}I\_vect_i\right)\geq(L-1)\sigma_I^{2}$$

4.4.2 Vector_Mem

I vettori in uscita dal modulo Window vengono memorizzati in delle memorie, indirizzate dal segnale *c_o_win* e che vengono scritte solo quando la finestra risulta valida. Infatti, il segnale *v_win* viene collegato direttamente all'ingresso che abilita la scrittura della memoria.

Ad ogni passo di stima vengono forniti al modulo i segnali *c_i_param* e *v_i_param*. Il

primo viene utilizzato come indirizzo di lettura delle memorie, ottenendo in uscita i campioni della finestra temporale appartenenti ad una determinata cella. I dati letti vengono poi campionati e sincronizzati con i segnali prima citati per essere usati come ingressi dal modulo successivo.

4.4.2 Array_Compiler

Questo blocco utilizza i vettori *I_vect* = $\underline{u}$ e *V_vect* = $\underline{y}$ per creare la matrice di dimensione *L x 5*

$$A=\Phi=\begin{bmatrix}\varphi(t-L+1)\\ \varphi(t-L+2)\\ \vdots\\ \varphi(t)\end{bmatrix}$$

con

$$\varphi(t)=[-y(t-1)\quad -y(t-2)\quad u(t)\quad u(t-1)\quad u(t-2)]$$

ed il vettore *b* = $\underline{y}$, di lunghezza *L*. In questo blocco sono presenti dei multiplexer, collegati agli ingressi, che indirizzano i segnali in modo opportuno per creare le righe della matrice *A*:

$$A(1)=\varphi(t-L+1)=[-y(t-L)\quad -y(t-L-1)\quad u(t-L+1)\quad u(t-L)\quad u(t-L-1)]$$
$$A(2)=\varphi(t-L+2)=[-y(t-L+1)\quad -y(t-L)\quad u(t-L+2)\quad u(t-L+1)\quad u(t-L)]$$
$$\vdots$$
$$A(L)=\varphi(t)=[-y(t-1)\quad -y(t-2)\quad u(t)\quad u(t-1)\quad u(t-2)]$$

Si può notare che, oltre agli *L* campioni campioni contenuti nella finestra [*(u(t), y(t)), ...,*

(u(t-L+1), y(t-L+1))], per completare le prime due righe della matrice *A* occorrono anche altri due campioni di tensione e di corrente appartenenti alla finestra precedente (*u(t-L)*, *u(t-L-1)*, *y(t-L)* e *y(t-L-1)*). Per questo motivo la lunghezza effettiva dei vettori *I_vect* e *V_vect* è pari a *L + 2*. In questa implementazione *L* può variare da un minimo di 5 ad un massimo di 30. Considerando la lunghezza massima, i vettori *I_vect* e *V_vect* conterranno 32 elementi, la matrice *A* avrà una dimensione pari a 30 x 5 e il vettore b conterrà 30 elementi.

4.4.2 QRD

Il modulo QRD ha come ingressi le righe della matrice *A* ed il vettore *b*. Esso esegue il seguente algoritmo di decomposizione QR:

```
function [R,d]=givensqr(A,b)
  R=A
  d=b
  for j=1:n
    for i=j+1:m
      [R(j,j:end),R(i,j:end),d(j,1),d(i,1)]=
      givens(R(j,j:end),R(i,j:end),d(j,1),d(i,1))
    end
  end
end
```

Questo algoritmo permette di ricavare direttamente la matrice *R* e il vettore *d*, senza calcolare la matrice *Q*, applicando la rotazione di Givens direttamente ai valori del vettore *b*. In questo modo si è ottenuto un notevole risparmio di risorse, sia perché si evitano tutte le operazioni necessarie all'ottenimento della matrice *Q* sia perché non è più necessario risolvere il sistema $d=Q^T b$.

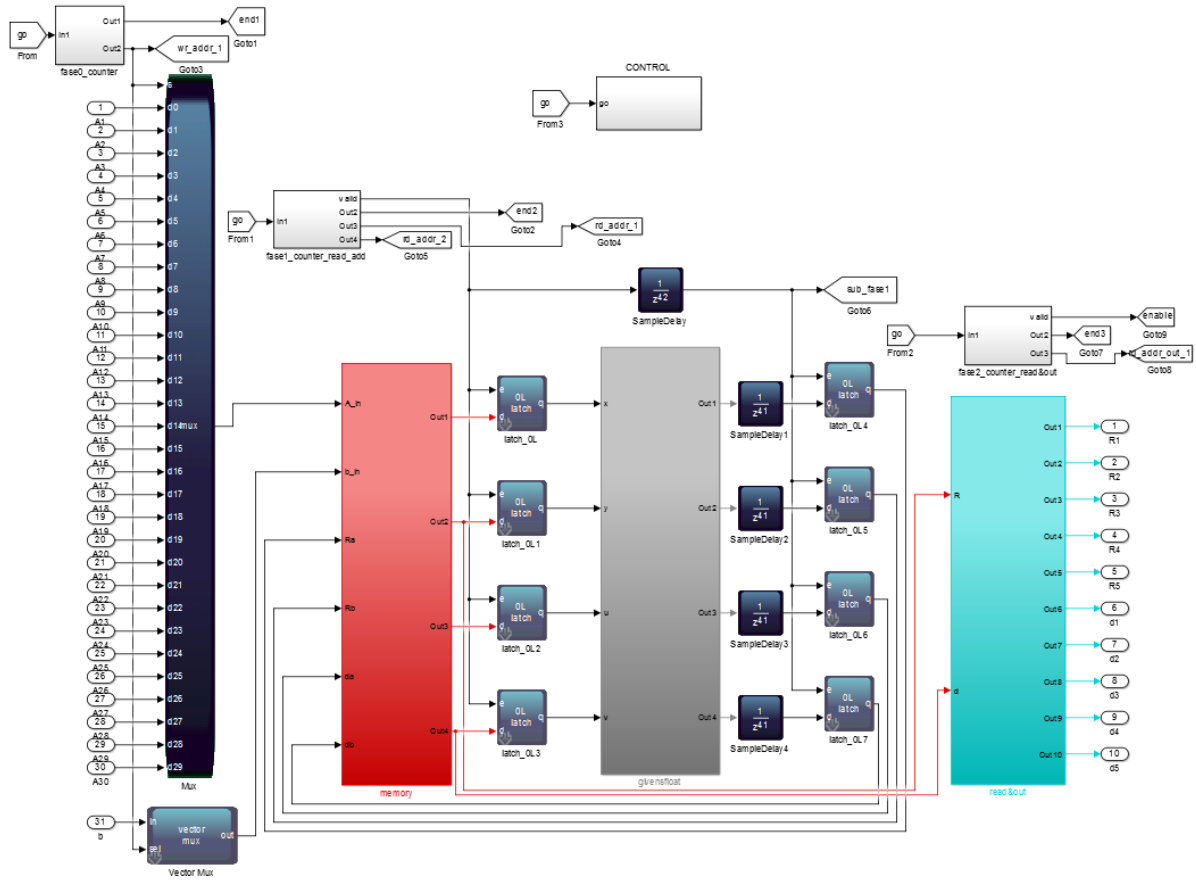

Figura 21: Modulo che effettua la decomposizione QR.

L'algoritmo può essere suddiviso in tre fasi, gestite dal blocco CONTROL e da altri blocchi, visibili in Figura 21, che hanno la funzione di generare in successione gli indirizzi necessari al corretto svolgimento dell'algoritmo ad ogni iterazione.

Nella prima fase si memorizzano gli ingressi in delle memorie situate nel blocco memory. In quest'ultimo, oltre ai blocchi di memorie del DSP Builder, è contenuta anche della logica che effettua il routing dei dati e dei segnali d'indirizzamento, in base alla fase in esecuzione. In una memoria viene memorizzata la matrice *A* e nell'altra il vettore *b*, le cui righe verranno sostituite rispettivamente con quelle della matrice *R* e del vettore *d* durante l'esecuzione. Questa fase equivale alle prime due uguaglianze (*R*=*A* e

d=*b*) dell'algoritmo riportato in precedenza.

Successivamente, si passa alla seconda fase, svolta principalmente dal blocco givensfloat, che esegue la funzione *givens*, vista nel precedente capitolo e riportata di seguito, in maniera più completa:

```
function [x,y,u,v]=givens(x,y,u,v)
  a=x(1); b=y(1)
  if b=0
    c=1; s=0
    return
  else
    if |b|>|a|
      t=-a/b; s=1/sqrt(1+t^2); c=st
    else
      t=-b/a; c=1/sqrt(1+t^2); s=ct
    end
  end
  x0=x; u0=u
  x(:)=cx0-sy; u(:)=cu0-sv
  y(:)=sx0+cy; v(:)=su0+cv
end
```

Ad ogni iterazione, questo blocco calcola i valori di *c* ed *s* e, successivamente, le righe di *R* e *d*, che verranno poi salvate in memoria e riutilizzate nelle successive iterazioni.

Per finire, nell'ultima fase, si leggono la matrice *R* ed il vettore *d* dalla memoria, si memorizzano le righe in un set di registri, contenuto nel blocco read&out, e si aggiornano le uscite che vengono fornite insieme ad un segnale di validazione generato dal blocco CONTROL.

4.4.2 Final_Calc

Una volta terminato l'algoritmo di decomposizione, si attiva il blocco Final_Calc,

suddiviso in due parti (Figura 22).

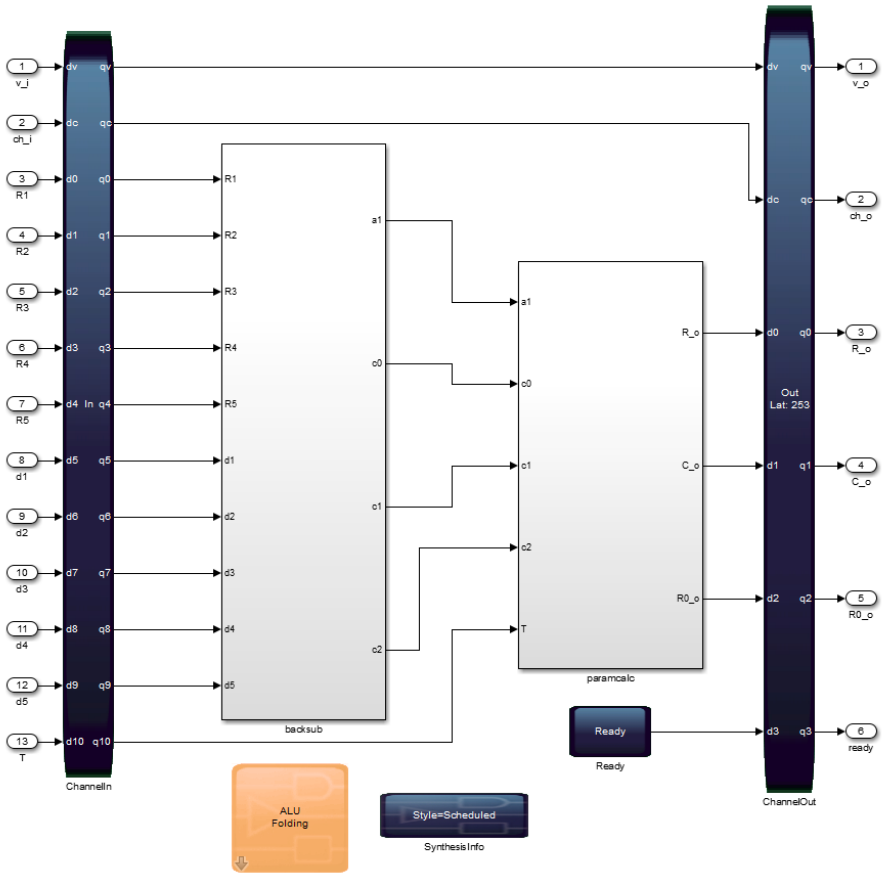

Figura 22: Modulo che effettua la sostituzione inversa e il calcolo dei parametri.

La prima parte (blocco backsub, Figura 23) si occupa risolvere il sistema $Rx = d$, utilizzando gli ingressi provenienti dal blocco precedente. In questo caso *x* è il vettore dei coefficienti da ricavare:

$$x = \begin{bmatrix} a_1 \\ a_2 \\ c_0 \\ c_1 \\ c_2 \end{bmatrix}$$

La matrice *R* è triangolare superiore e di dimensione *5 x 5* e quindi il sistema può

essere risolto con l'algoritmo della back substitution (sostituzione inversa), riportato di seguito, in codice MATLAB-like:

$x=[0;0;0;0;d(k,1)/R(k,k)]$
for $k=n-1:-1:1$

$$x(k,1)=\frac{d(k,1)-R(k,k+1:n)x(k+1:n,1)}{R(k,k)}$$

end

Ogni passo d'iterazione viene eseguito da un blocco diverso, come si nota in Figura 23, che ha come ingresso una riga della matrice *R*, la stessa riga del vettore *d* e i valori calcolati dai blocchi precedenti.

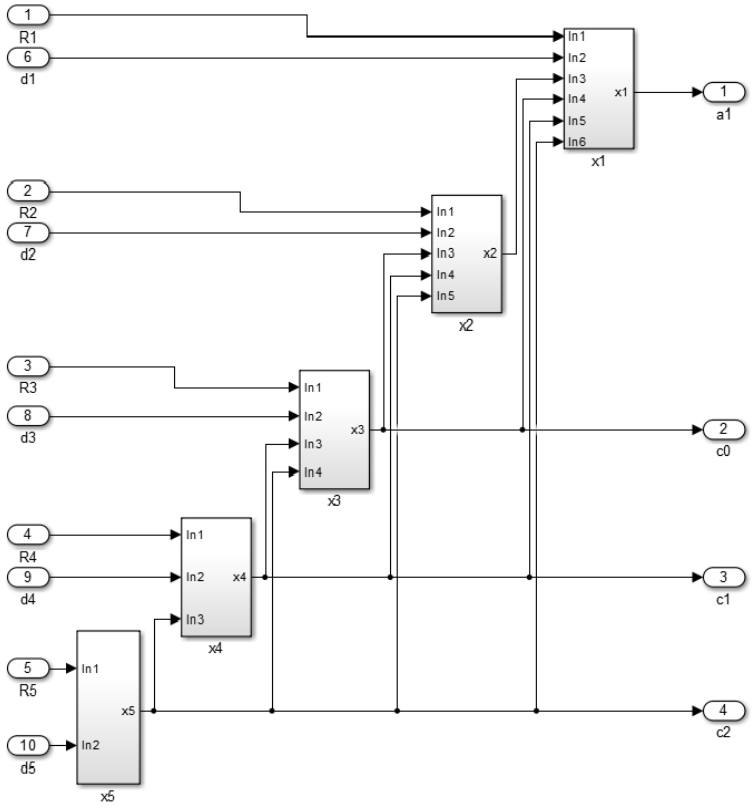

Figura 23: Modulo che esegue la sostituzione inversa.

Questo non vale per il blocco x5 che è formato da un semplice divisore. Il valore del coefficiente a_2 non viene riportato in uscita poiché non viene utilizzato dal blocco successivo.

Determinati i coefficienti {a_1, c_0, c_1, c_2}, il blocco paramcalc (Figura 24) si occupa di calcolare il valore dei parametri {R_0, R, C} dalle equazioni univoche ricavate nel paragrafo 3.4.1 e riportate di seguito per completezza:

$$R=-\frac{T}{2C}\frac{a_1}{2+a_1}$$

$$\begin{cases} X1=T^2\left(-1+\frac{a_1}{2+a_1}\right)b_1+2Q_RT\left(-1+\frac{a_1}{2+a_1}\right)R_0+Q_RT^2\left(\frac{a_1}{2+a_1}\right)\frac{1}{C} \\ X2=-2T^2b_1-4Q_RT\left(\frac{a_1}{2+a_1}\right)R_0 \\ X3=-T^2\left(1+\frac{a_1}{2+a_1}\right)b_1+2Q_RT\left(1+\frac{a_1}{2+a_1}\right)R_0-Q_RT^2\left(\frac{a_1}{2+a_1}\right)\frac{1}{C} \end{cases}$$

dove

$$X1=2Q_RT\left(1-\frac{a_1}{2+a_1}\right)c_0$$

$$X2=2Q_RT\left(1-\frac{a_1}{2+a_1}\right)c_1$$

$$X3=2Q_RT\left(1-\frac{a_1}{2+a_1}\right)c_2$$

Risolvendo il sistema e raggruppando i termini in comune si ottengono delle equazioni

in forma compatta che permettono di realizzare questo calcolo senza l'utilizzo di eccessive risorse. In particolare, si ottengono le seguenti relazioni:

$$R_0 = \frac{v}{4t}(c_0 + c_2 - c_1)$$

$$R = -\frac{1}{2x}$$

$$C = t\,T\,x$$

dove:

$$t = \frac{a_1}{2 + a_1} \quad ; \quad v = 1 - t \quad ; \quad x = \frac{2t}{v[(c_0 + c_2)(1 + t^2) + (c_0 - c_2)2t - (1 - t^2)c_1]}$$

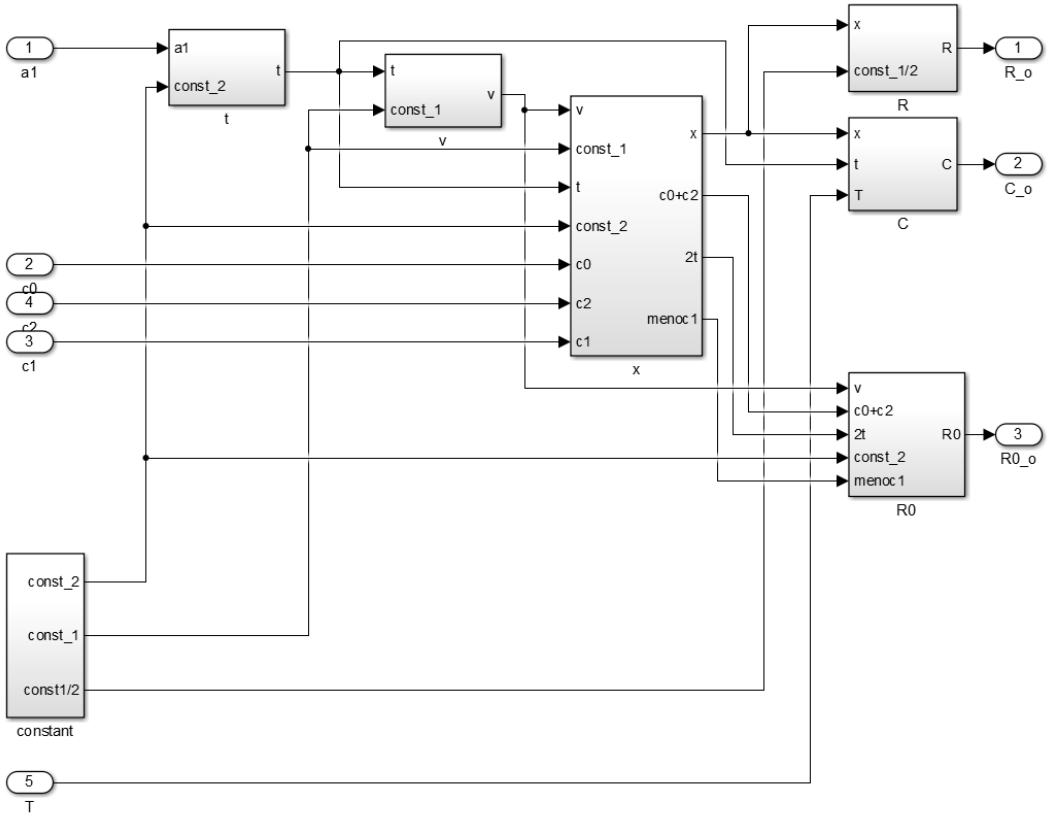

Figura 24: Modulo per il calcolo dei parametri.

In Figura 24 è riportata l'implementazione del calcolo, realizzata utilizzando le solite librerie del DSP Builder.

Terminato anche questo calcolo, viene fornito un segnale di valid in uscita (*v_o*) che permette la memorizzazione dei nuovi parametri calcolati, che verranno utilizzati dal modulo di stima dello stato di carica per la stima successiva.

5 Validazione dell'implementazione dell'algoritmo

In questo capitolo viene presentata la validazione dell'implementazione dell'algoritmo su FPGA. Viene quindi descritta la scheda di sviluppo utilizzata e l'FPGA di cui è dotata, la metodologia con la quale è stato realizzato il test ed infine vengono riportati i risultati ottenuti in confronto con quelli relativi alle simulazioni eseguite in ambiente Simulink.

5.1 Descrizione generale del test

Terminata l'implementazione dell'algoritmo, seguendo il flusso di progetto presentato nel precedente capitolo (Figura 10), si può compilare il sistema con Quartus II ed effettuare una verifica su FPGA per testare il funzionamento dell'algoritmo. Questa verifica è necessaria per confermare la validità dello strumento di sintesi e del flusso di progetto adottato.

Il sistema viene quindi implementato su un FPGA Altera appartenente alla famiglia Cyclone V, presente sulla scheda di sviluppo SoCKit. Per eseguire il test vengono utilizzati campioni di tensione e di corrente provenienti da misurazioni effettuate durante test precedentemente realizzati su una cella Kokam SLPB723870H4. È stato quindi necessario caratterizzare la cella ricavando la caratteristica SOC-OCV e il valore dei parametri del modello con cui inizializzare l'algoritmo. I campioni sono stati ricavati applicando alla cella un particolare profilo di corrente utile a testare il comportamento in situazioni in cui la cella opera su un veicolo elettrico. Essi vengono inviati al sistema tramite un Virtual Instrument di LabVIEW che si occupa anche della memorizzazione

dei risultati dei calcoli. I risultati verranno quindi confrontati con quelli derivanti dalle simulazioni realizzate in ambiente Simulink.

5.2 Scheda di sviluppo SoCKit

La scheda di sviluppo SoCKit è una piattaforma hardware realizzata intorno ad un SoC (System-on-Chip) FPGA di Altera che combina un processore ARM hard-core Cortex A9 (Hard Processor System, HPS) alla logica programmabile.

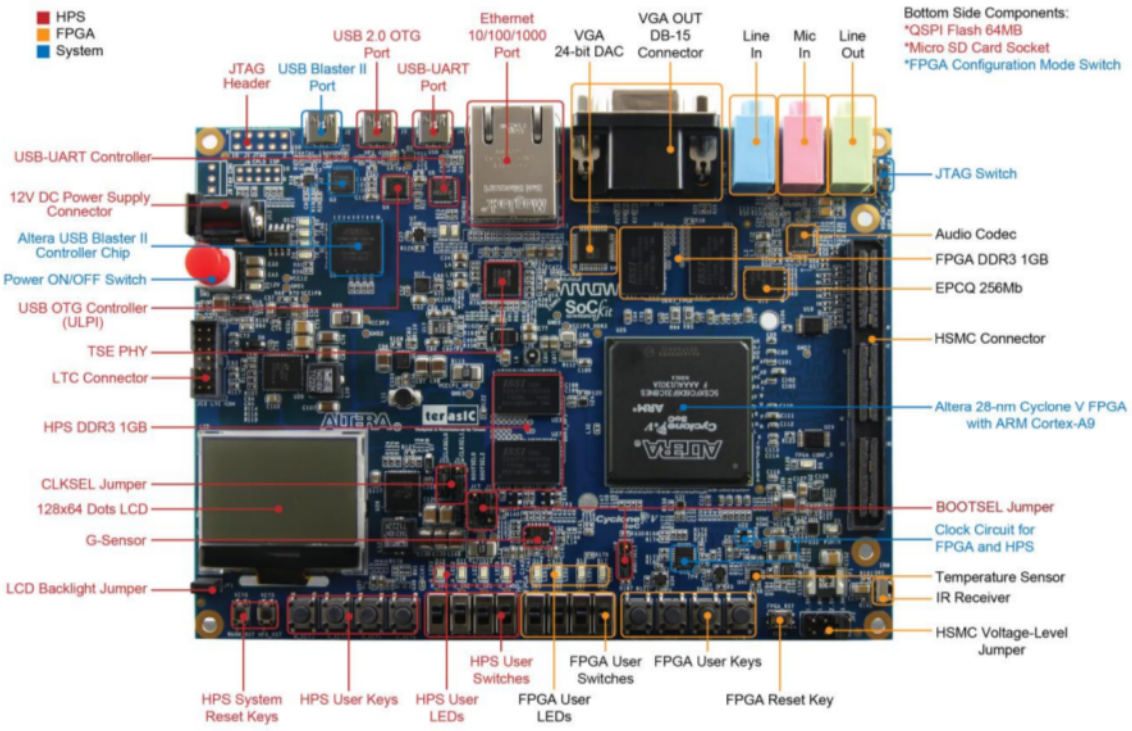

Figura 25: SoCKit Development Kit.

La scheda include numerose periferiche hardware come banchi di memoria DDR3, codec audio e video, socket per schede micro SD, comunicazione seriale USB-UART, connettore LTC con comunicazione seriale SPI o I2C, chip Ethernet e molto altro

(Figura 25).

Sulla scheda è presente un dispositivo Altera Cyclone V SoC 5CSXFC6D6F31, un FPGA relativamente a basso costo e a basso consumo di potenza, ma utilizzabile per realizzare applicazioni anche molto complesse idonee per ambiti industriali e automotive. Il dispositivo presenta le seguenti caratteristiche:

- 110000 elementi logici (LE);
- 41910 adaptive logic module (ALM);
- 166036 registri;
- 5662720 bit di memoria;
- 112 variable-precision digital signal processing (DSP) block;
- 15 PLL;
- HPS ARM Cortex-A9 dual core.

Gli Adaptive Logic Module (ALM) sono i blocchi base dell'FPGA che possono essere configurati per implementare funzioni logiche, aritmetiche o di memorizzazione. Essi sono costituiti da:

- una LUT partizionabile a otto ingressi che può essere configurata in vari modi. Grazie ad essa è possibile implementare tutte le funzioni logiche a 6 ingressi, alcune funzioni a 7 ingressi oppure può essere divisa in LUT indipendenti più piccole, come ad esempio due LUT a 4 ingressi;
- due sommatori dedicati che permettono di effettuare due somme a 2 bit o due somme a 3 bit, senza l'utilizzo di logica aggiuntiva;
- quattro registri, per permettere la facile implementazione di funzioni che necessitano di molti registri o di effettuare il pipeling per risolvere le

problematiche di timing ed ottimizzare le prestazioni.

Il software Quartus II configura automaticamente ogni ALM per ottimizzare le performance, l'efficienza, il consumo di potenza e l'area. Un quarto degli ALM totali presenti nel dispositivo può essere configurato per essere utilizzato come memoria, ottimizzata per la realizzazione di linee di ritardo per i filtri, piccoli buffer FIFO e shift register.

Numerosi vantaggi derivano anche dall'impiego dei variable-precision Digital Signal Processing (DSP) block, in quanto offrono il supporto per moltiplicazioni a diverse precisioni e integrano delle caratteristiche per migliorare le prestazioni di funzioni comuni nell'elaborazione dei segnali digitali, come ad esempio i filtri FIR. I DSP block offrono le seguenti funzionalità:

- esecuzione di moltiplicazioni con numero variabile di bit (9, 18 e 27 bit), con elevate prestazioni e ottimizzate dal punto di vista del consumo di potenza;
- possibilità di realizzare moltiplicazioni tra numeri complessi;
- possibilità di effettuare la somma, la sottrazione e l'accumulazione per combinare i risultati delle moltiplicazioni, all'interno dello stesso blocco;
- collegamento in cascata per formare linee di ritardo utili per applicazioni di filtraggio;
- propagazione dell'uscita verso altri blocchi senza l'utilizzo di logica esterna;
- integrano pre-adder e banchi di registri interni per la memorizzazione di coefficienti da utilizzare nelle moltiplicazioni, sempre per semplificare l'implementazione di filtri digitali;
- supporto per l'aritmetica floating point, single o double precision.

5.2.1 Occupazione delle risorse dell'FPGA

L'FPGA disponibile risulta idonea all'applicazione in questione in quanto le operazioni in floating point, necessarie all'esecuzione dell'algoritmo, vengono implementate con l'ausilio dei DSP block, consentendo un risparmio notevole di elementi logici. Si ha quindi la possibilità di poter sviluppare e migliorare ulteriormente l'algoritmo o di aggiungere ulteriori funzionalità al sistema.

Nella seguente tabella vengono riportate le informazioni più importanti sull'occupazione delle risorse del sistema.

Sistema hardware	ALMs usati/disponibili		DSP blocks usati/disponibili		Memory bits usati/disponibili	
Stima del SOC	2274/41910	5%	5/112	4%	66592/5662720	1%
Identificazione online dei parametri	15081/41910	36%	37/112	33%	164305/5662720	3%
Sistema complessivo	21588/41910	52%	42/112	38%	796945/5662720	14%

Tabella 4: Impiego delle risorse del sistema per un FPGA Altera Cyclone V SoC 5CSXFC6D6F31.

Nel sistema complessivo, oltre ai moduli hardware realizzati con DSP Builder, è compreso anche il modulo Memory Mapped che si occupa di far comunicare tra loro i due blocchi e dell'interfacciamento con l'HPS. In particolare, memorizza gli ingressi e le uscite in un set di registri accessibili dal processore hardware e si occupa di fornire i segnali di abilitazione ai moduli, eseguendo anche le procedure necessarie per effettuare il time multiplexing tra le celle.

Si nota che l'occupazione complessiva è di circa la metà delle risorse disponibili nel dispositivo. Questo permette di implementare il sistema su un dispositivo che dispone di

meno risorse e quindi più economico o di poter sviluppare ulteriori moduli hardware che eseguano algoritmi utili a stimare in maniera più precisa lo stato della batteria, come ad esempio un algoritmo di stima dello stato di salute.

5.3 Descrizione del sistema di verifica

Per poter verificare il corretto funzionamento del sistema implementato è stato sviluppato un setup di verifica preliminare molto semplice (Figura 26).

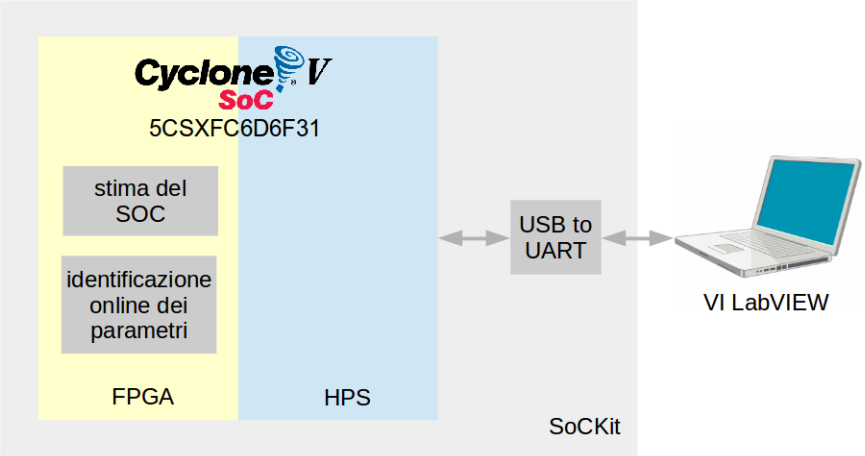

Figura 26: Struttura generale del sistema di verifica.

L'HPS viene usato, in fase di verifica, per fornire ai moduli hardware i campioni di tensione e di corrente. Questi campioni vengono inviati al sistema tramite un PC, su cui è in esecuzione un Virtual Instrument (VI) di LabVIEW. Inoltre, l'HPS invia a quest'ultimo i risultati del calcolo per permetterne la memorizzazione. La comunicazione avviene tramite interfaccia seriale, utilizzando una periferica disponibile sulla scheda di sviluppo e direttamente collegata all'HPS. Dunque è stato realizzato un firmware che ha le

seguenti funzionalità:

- inizializzazione del sistema e delle periferiche;
- ricezione, da parte del VI, dei parametri di inizializzazione e dei campioni da fornire all'algoritmo durante il test;
- scrittura ciclica dei valori ricevuti negli appositi registri dell'interfaccia Memory Mapped e abilitazione del calcolo;
- lettura dei risultati ottenuti e invio dei dati al VI.

Il VI di LabVIEW (Figura 27) permette l'invio, la memorizzazione e la visualizzazione dei dati e non effettua su di essi nessuna operazione o calcolo.

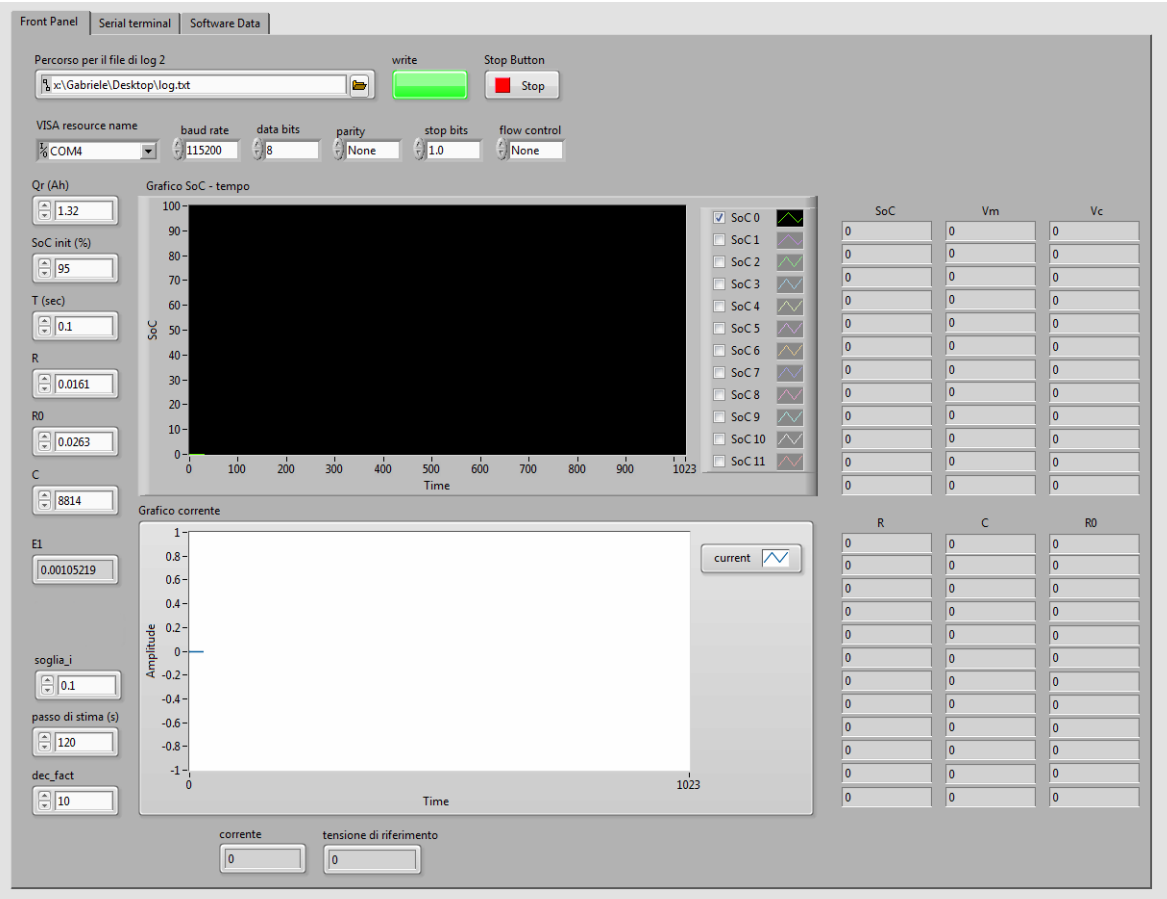

Figura 27: Front panel del VI di LabVIEW.

Il Virtual Instrument realizzato ha le seguenti funzioni:

- configurazione della comunicazione seriale, specificando la porta di comunicazione (COM) e altri parametri come baud rate, numero di bit per ogni dato, la presenza del bit di parità, ecc.;
- scelta del valore dei parametri di inizializzazione da inviare al modulo, quali R, R0, C, SOCinit, T, Qr, il valore della soglia per la verifica della validità della finestra temporale, il passo di stima ed il fattore di decimazione;
- scelta del percorso del file testuale di log nel quale verranno memorizzati i risultati;
- visualizzazione dei dati ricevuti per verificare il corretto andamento del test durante l'esecuzione dello stesso.

Da notare che il VI è predisposto per ricevere i valori del calcolo relativi ad una serie di 12 celle, ma per verificare il corretto funzionamento dell'algoritmo il test in questione viene eseguito simulando una singola cella.

5.4 Caratterizzazione della batteria

Per lo sviluppo dell'algoritmo e la sua validazione è stata utilizzata una cella di riferimento. In particolare, si è scelto di usare una cella Kokam SLPB723870H4, una batteria NMC ultra high power da 1.5 Ah, le cui caratteristiche principali sono riassunte in Tabella 5.

Per ricavare la caratteristica OCV-SOC e i parametri del modello, utili nei test di calcolo del SOC a parametri costanti, relativi alla cella usata, sono stati effettuati dei test di misura, descritti nei successivi paragrafi.

Specifiche tecniche		
Capacità nominale		1.5 Ah
Tensione nominale		3.7 V
Carica	Corrente massima	1.5 A
	Tensione massima	4.2 V
Scarica	Corrente massima continua	30.0 A
	Corrente massima impulsata	60.0 A
	Tensione di cut-off	2.7 V
Cicli totali di scarica		> 500
Temperatura operativa	Carica	0 – 40 °C
	Scarica	-20 – 60 °C

Tabella 5: Specifiche della cella Kokam SLPB723870H4 [29].

5.4.1 Caratteristica OCV-SOC

La caratteristica OCV-SOC è necessaria alla realizzazione dell'algoritmo di stima del SOC in quanto permette di calcolare la tensione di uscita del modello. Questa caratteristica può essere ricavata grazie a dei test composti da cicli di carica e scarica.

Per misurare la OCV occorrono lunghi tempi di rilassamento (da trenta minuti ad un'ora per la cella in esame) nei quali il carico viene scollegato (la corrente è quindi nulla) per permettere alla tensione di cella di andare a regime. Per ottenere la caratteristica desiderata occorrono diverse misure e per questo motivo test di questo tipo sono molto lunghi. Per ovviare a questo problema si può pensare di eseguire un certo numero di misure per poi interpolare i dati.

Il test viene effettuato con l'ausilio di Keithley 2420, un'unità source-meter in grado di funzionare sia da carico/sorgente per la cella che di misurare la sua corrente e tensione. Si effettua quindi una scarica completa, seguita da una carica, ad intervalli del

5% della capacità nominale Q_n (quindi ogni impulso di corrente è di $I_0 = 1.5\,A$ e ha una durata tale da determinare la variazione di SOC desiderata, cioè tre minuti), separati da una pausa di rilassamento di mezzora, durante la quale la tensione rilassa con un andamento esponenziale verso il valore di OCV, che si suppone sia raggiunto alla fine della pausa (Figura 28). Tra un ciclo completo ed il successivo è presente una pausa di un'ora.

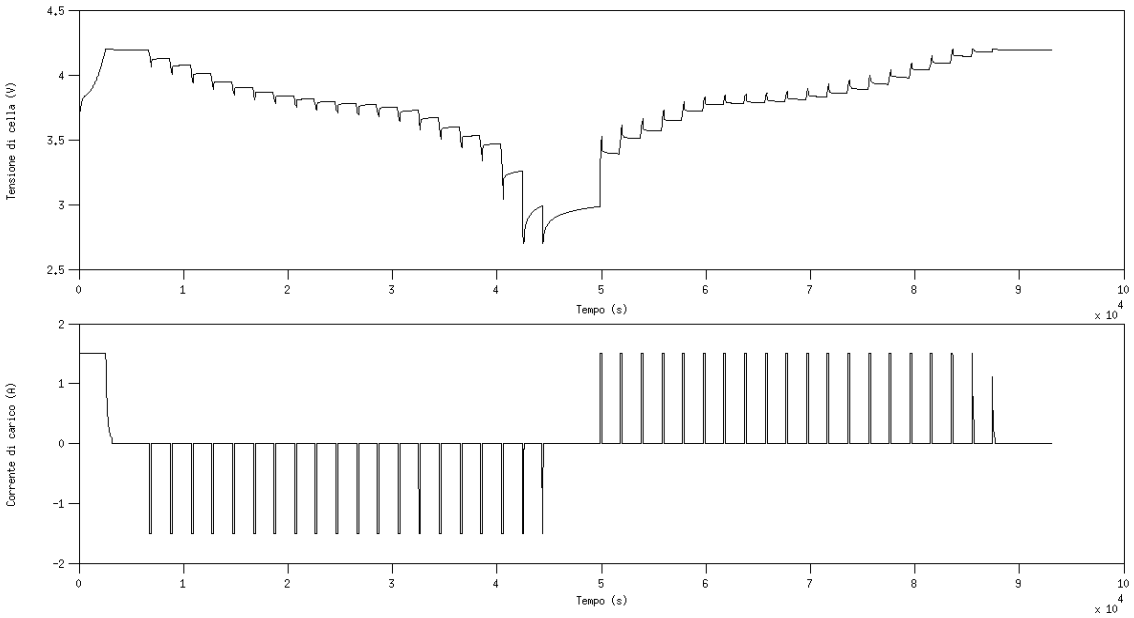

Figura 28: Andamento della tensione e della corrente durante il test.

Il valore di SOC corrispondente ad ogni misura si ricava invece come rapporto tra la carica erogata/assorbita dalla cella (ricavata tramite integrazione della corrente) e la sua carica estratta totale.

In questo modo si ottengono venti punti in carica e venti in scarica che verranno poi interpolati in MATLAB per ottenere la caratteristica desiderata (Figura 29). Si può notare

che il fenomeno di isteresi, per celle di questo tipo, è poco significativo e può quindi essere trascurato, utilizzando come caratteristica OCV-SOC la media tra la curva in carica e quella in scarica.

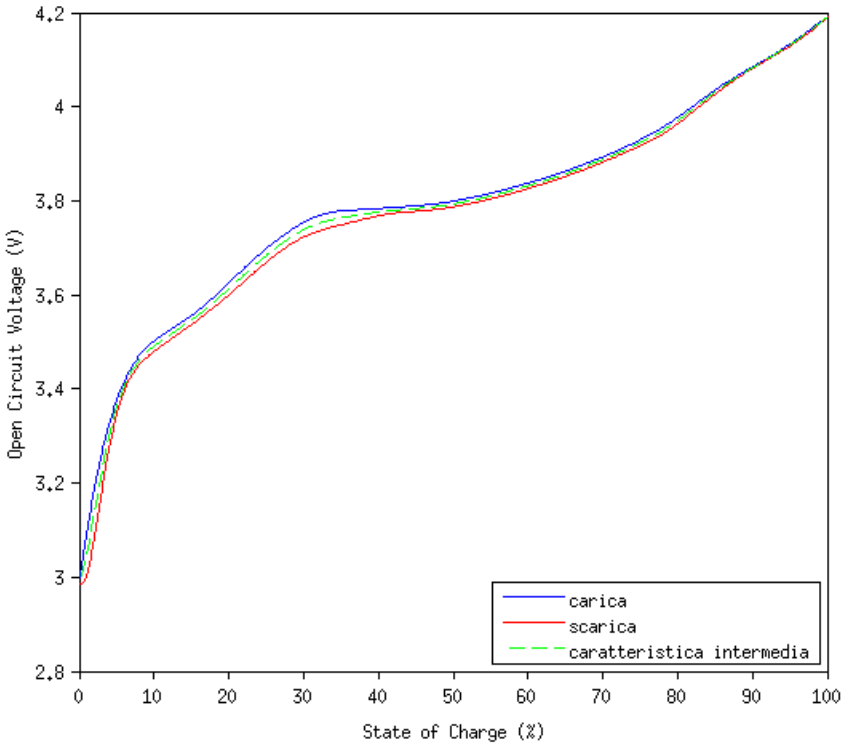

Figura 29: Caratteristica OCV-SOC.

5.4.2 Stima offline dei parametri

La stima online dei parametri avviene con un determinato passo di stima. Prima che venga effettuato il primo calcolo occorre utilizzare dei valori noti dei parametri. Per questo motivo è necessaria una misura offline dei parametri del modello di cella. Questa misura può essere realizzata con lo stesso test descritto nel precedente paragrafo, in quanto occorre analizzare l'andamento della tensione in condizioni di

carica, scarica e di rilassamento. Questo deve avvenire per diversi valori del SOC visto che i parametri dipendono dallo stato di carica.

Considerando quindi il modello ridotto di Figura 7 e ricordando che

$$\begin{cases} \dfrac{d}{dt} v_{RC}(t) = -\dfrac{v_{RC}(t)}{RC} + \dfrac{i_L(t)}{C} \\ v_T(t) = V_{OC} - R_0 i_L(t) - v_{RC}(t) \end{cases}$$

per effettuare l'estrazione dei parametri si può confrontare la tensione di uscita del modello, in risposta ad una corrente impulsata a gradini come quella in Figura 28, con la tensione reale misurata.

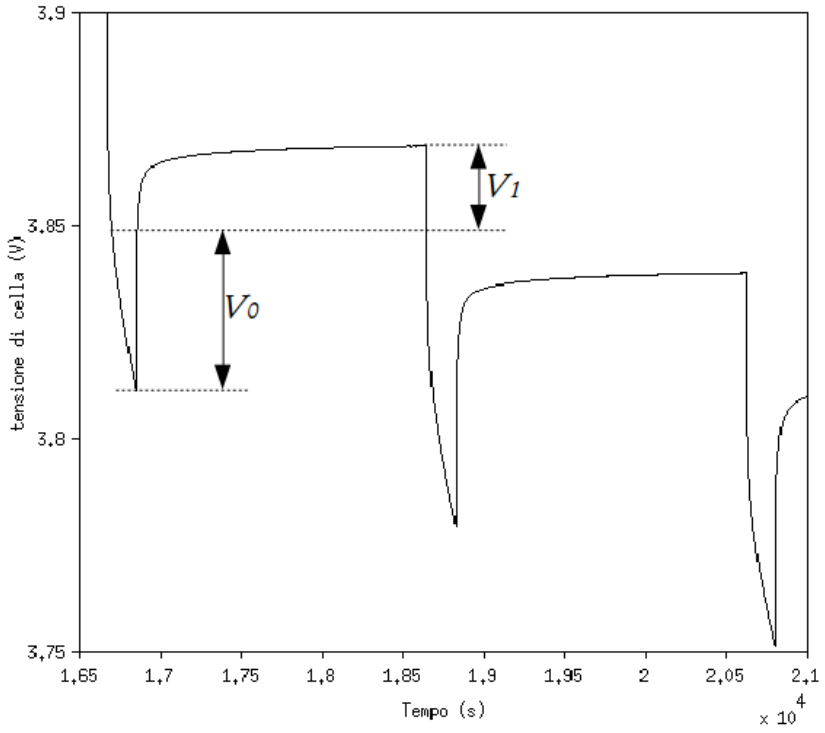

Figura 30: Andamento della tensione di cella durante i cicli di scarica.

L'andamento della tensione può essere diviso in due fasi, visibili in Figura 30, nella quale viene rappresentato un tratto dell'andamento della tensione durante la procedura di scarica. Quando si passa da una condizione di scarica ad una di rilassamento la tensione presenta un tratto verticale causato dal cambiamento discontinuo della corrente che passa da I_0 a 0 e, successivamente, un tratto esponenziale dovuto al rilassamento e riprodotto, nel modello, dal gruppo RC.

L'ampiezza del salto ha un valore pari a $V_0 = R_0 \cdot I_0$ e quindi misurandolo e dividendolo per I_0, si ottiene il valore di R_0. La componente esponenziale ha un andamento pari a $V_1\left(1-e^{-t/\tau}\right)$ e, utilizzando la funzione di fitting esponenziale di MATLAB, è possibile ricavare i valori di V_1 e τ. Dunque si determinano R e C sapendo che $R = V_1 \cdot I_0$ e che $C = \tau / R$.

L'andamento dei parametri durante il test è riportato in Figura 31. I valori medi dei parametri calcolati corrispondono a:

- $R_0 = 0.0263\,\Omega$
- $R = 0.0161\,\Omega$
- $C = 8814\,F$

e vengono utilizzati per inizializzare l'algoritmo di stima.

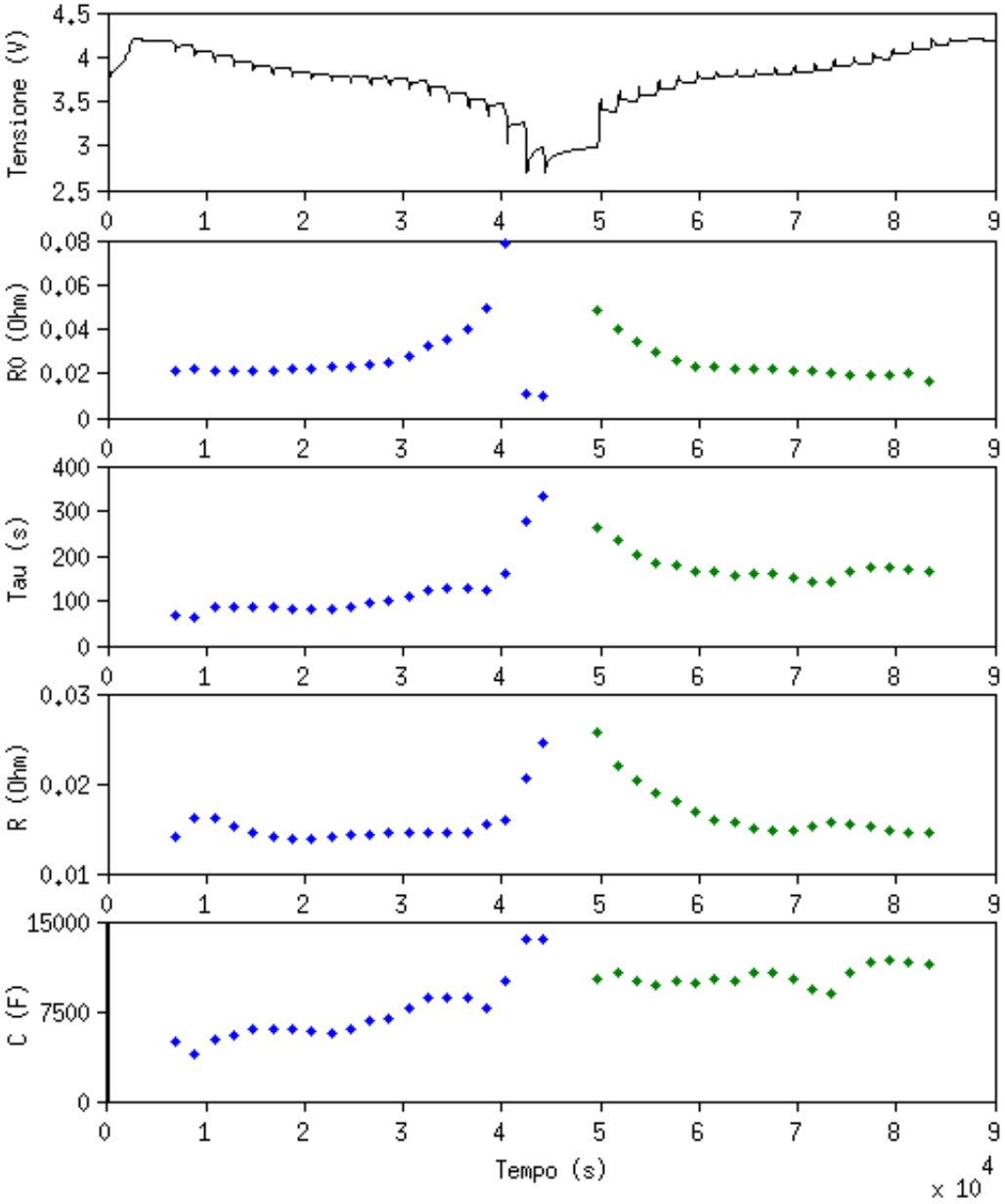

Figura 31: Grafico dei valori dei parametri calcolati.

5.5 Urban Dynamometer Driving Schedule (UDDS)

I campioni di corrente e tensione utilizzati provengono da un test precedentemente eseguito in cui è stato utilizzato un particolare profilo di corrente per testare il

funzionamento dell'algoritmo in particolari condizioni di utilizzo.

L'algoritmo sviluppato è pensato per le batterie al litio, soprattutto nell'ottica di un impiego in applicazioni automotive. I test effettuati devono garantire una buona valutazione dell'algoritmo implementato nelle condizioni operative della cella quando essa alimenta un veicolo elettrico. Una possibile scelta è quella di utilizzare dei cicli di guida standard, largamente utilizzati per testare il motore a combustione interna delle auto, valutandone i consumi e le emissioni. In questo caso si è scelto di utilizzare lo Urban Dynamometer Driving Schedule (UDDS), definito dalla U.S. Environmental Protection Agency [30]. Un ciclo UDDS simula un percorso urbano di lunghezza pari a 12.07 Km e della durata di 22 minuti con numerose fermate.

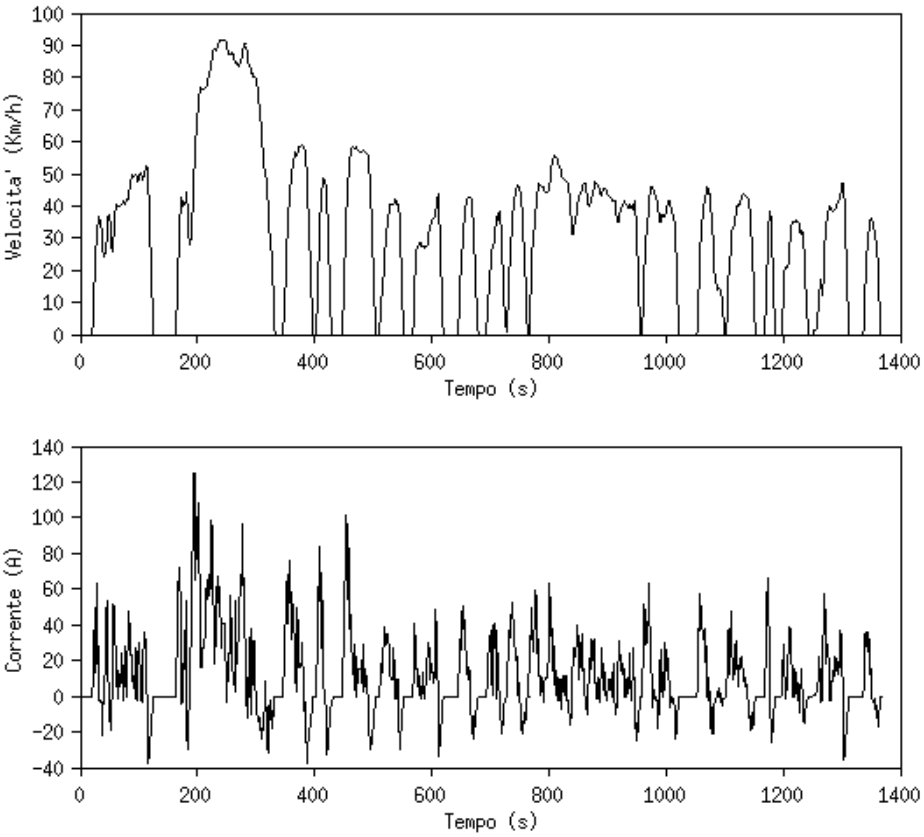

Figura 32: Profilo di velocità e corrente dell'UDDS.

La velocità media del veicolo è di 31.5 Km/h con un picco massimo di 91.85 Km/h. In Figura 32 è riportato il profilo di velocità e il corrispondente profilo di corrente, ricavato seguendo la procedura riportata in [4], per una batteria da 66.2 Ah.

Adattando il valore della corrente in proporzione alla capacità della cella in uso (pari a 1.5 Ah) è possibile utilizzare il profilo ottenuto, ripetendolo per più cicli, per effettuare i test desiderati, al fine di validare l'algoritmo.

5.6 Risultati del test

I campioni utilizzati sono stati ricavati da un test in cui sono stati applicati alla cella un certo numero di profili UDDS di corrente. Questo profilo è stato ricostruito come una funzione a scalini in cui ogni gradino corrisponde ad un valore di corrente che il Keithley 2420 applica alla batteria. Durante l'esecuzione del test il Keithley 2420 viene utilizzato come carico e generatore; inoltre, misura e memorizza il valore della corrente di cella e della tensione ai suoi capi.

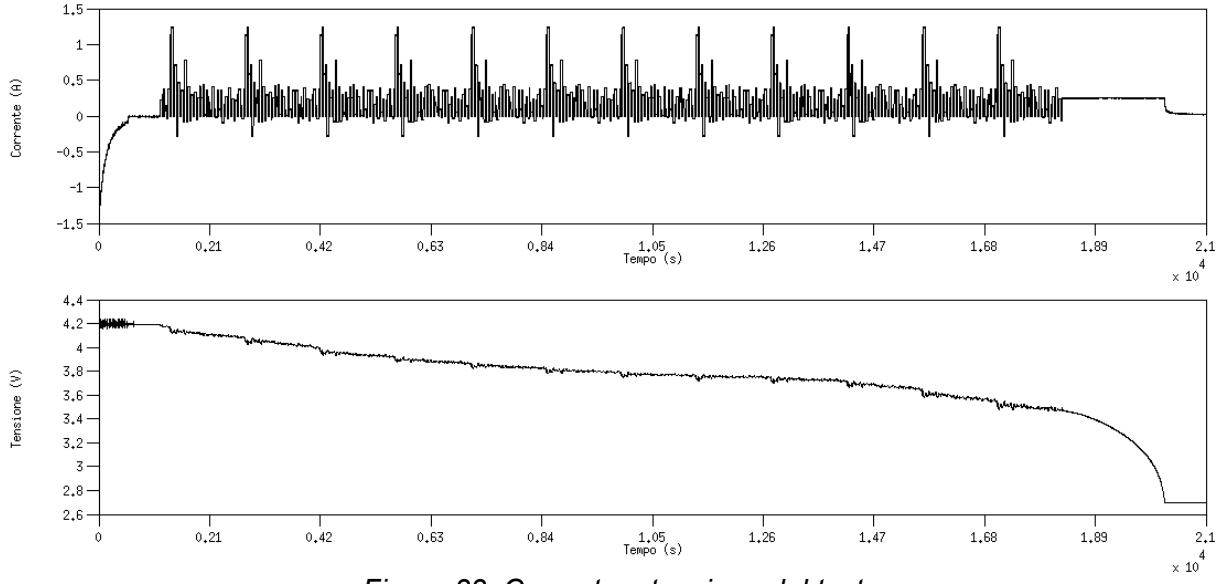

Figura 33: Corrente e tensione del test.

Il test è composto dalle seguenti fasi, visibili in Figura 33:

- carica completa iniziale fino al 100% del SOC, per portarsi in uno stato noto;
- applicazione di 12 cicli UDDS consecutivi che permettono di scaricare la cella fino al 10% del SOC;
- scarica completa a corrente costante per raggiungere lo 0% del SOC.

I campioni di corrente e tensione, acquisiti con un periodo di campionamento pari a 0.1 s, sono stati memorizzati in un file testuale che viene letto dal VI di LabVIEW e inviati all'HPS. Ad ogni ciclo i campioni ricevuti vengono forniti ai moduli hardware ed elaborati. I risultati vengono nuovamente inviati al VI per consentirne la memorizzazione.

Durante il test la comunicazione tra i due moduli hardware (stima del SOC e identificazione online dei parametri) è stata interrotta in modo da poterne verificare il funzionamento singolarmente.

5.6.1 Modulo di stima del SOC

Il modulo di stima del SOC esegue il calcolo utilizzando i valori costanti di inizializzazione dei parametri. Il confronto tra i valori di SOC ottenuti dalla simulazione in Simulink e quelli ottenuti dal modulo hardware sono riportati in Figura 34. Questi andamenti sono relativi solo alla parte centrale del test, durante l'esecuzione dei 12 cicli UDDS. Si può notare che i risultati del test sono del tutto coerenti con quelli ottenuti dalle simulazioni. Infatti, nell'ingrandimento in Figura 35 si vede che i due andamenti, rappresentati in colore nero e blu, sono sovrapposti.

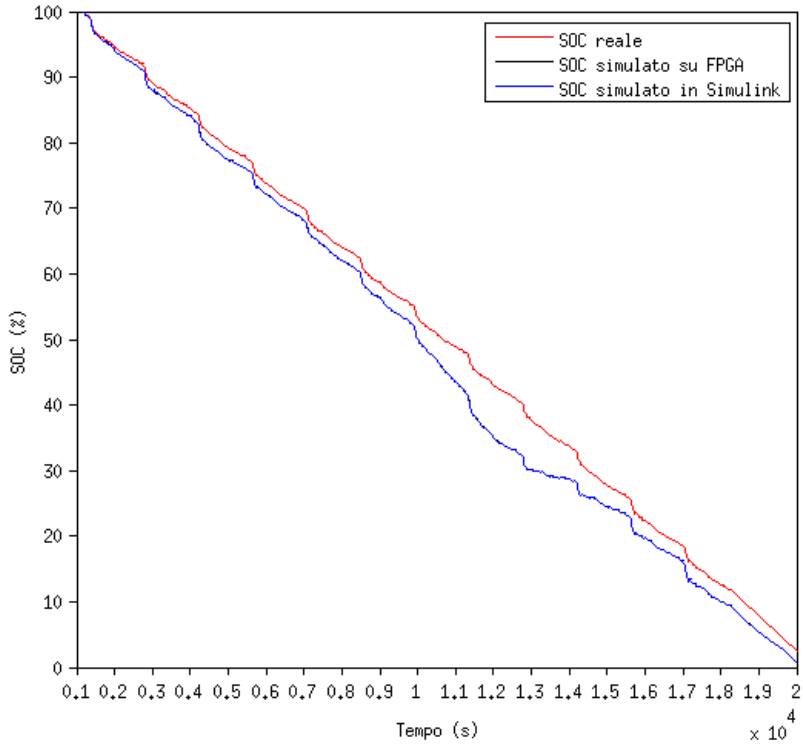

Figura 34: Confronto tra i risultati simulati su FPGA e su Simulink.

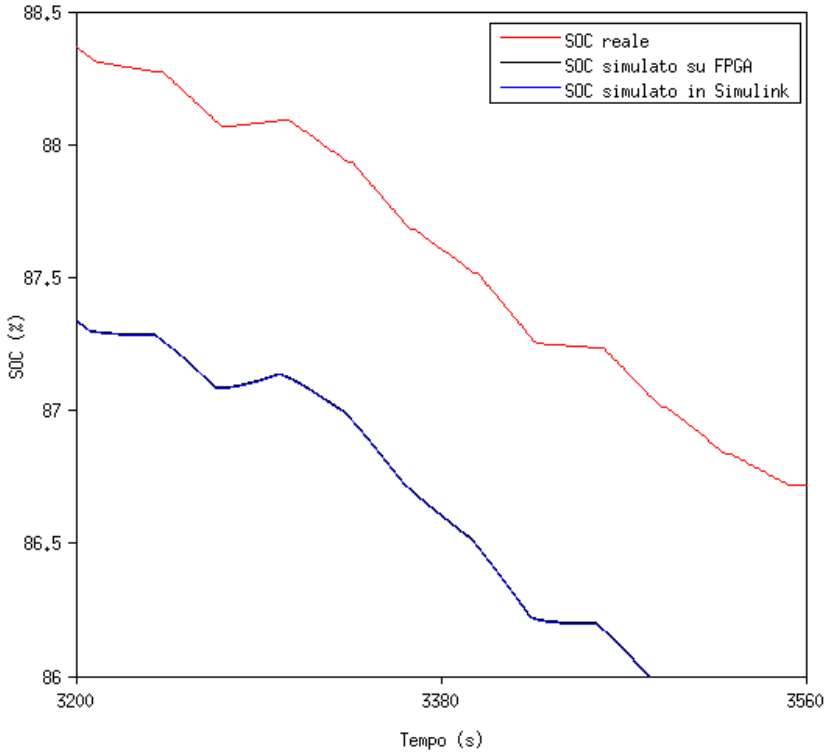

Figura 35: Ingrandimento del confronto tra i risultati simulati su FPGA e su Simulink.

Nelle figure appena citate è riportato anche l'andamento del SOC reale usato come riferimento in fase di sviluppo e test dell'algoritmo. Esso è stato ottenuto applicando la tecnica del Coulomb Counting, integrando direttamente la corrente misurata dal Keithley 2420:

$$SOC(t) = SOC_{init} - \frac{1}{Q_R}\int_{t_0}^{t} i_L(\tau)\, d\tau$$

La precisione di questo risultato è molto elevata in quanto la carica completa, effettuata all'inizio del test, permette di partire da uno stato noto e quindi di conoscere con precisione il valore iniziale del SOC (SOC_{init}); inoltre, scaricando tutta la cella ed integrando la corrente si ottiene il valore reale della carica massima estraibile (Q_R), diversa dalla capacità nominale a causa dell'invecchiamento e delle condizioni operative.

Come si nota, nella stima a parametri costanti, il SOC segue relativamente bene l'andamento reale ad eccezione della zona di valori del SOC intorno al 40%-50%. L'errore commesso ($SOC_{err} = SOC_{reale} - SOC_{stimato}$) può essere quantificato utilizzando il valore RMS dell'errore (Root Mean Square Error, RMSE), definito dalla relazione:

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N} SOC_{err}^2(k)}$$

e il suo valore massimo, definito come:

$$E_{max} = \max_k |SOC_{err}(k)|$$

In questo caso si ha che $RMSE = 3.9953$ e che $E_{max} = 9.8513$. Questi errori sono dovuti a essenzialmente a due fattori:

- il modello non rappresenta correttamente il comportamento della cella;
- la riduzione del tempo di risposta del sistema, dovuta al fatto che la caratteristica OCV-SOC, per valori di SOC intorno al 40%, ha una pendenza molto bassa.

Il primo fattore è dovuto essenzialmente al fatto che i parametri del modello sono costanti nel tempo e quindi non è specificata la loro dipendenza dallo stato della batteria durante la sua vita. Per quanto riguarda il secondo fattore si può dimostrare che la costante di tempo del sistema dipende dalla pendenza della caratteristica OCV-SOC [22], in particolare si ha che:

$$\tau_s = \frac{Q_R (R_0 + R)}{b_1}$$

dove b_1 è la pendenza della curva. Se essa diminuisce la costante di tempo del sistema aumenta e se questo incremento avviene in maniera considerevole la tensione di uscita del modello non riesce a seguire la tensione reale della cella. Su tempi molto lunghi il sistema funzionerebbe comunque in maniera corretta, ma nel caso in questione, in cui il SOC varia sensibilmente a causa del profilo di corrente adottato, si crea un errore abbastanza elevato proprio in corrispondenza della zona “piatta” della caratteristica.

5.6.2 Modulo di identificazione online dei parametri

La stessa procedura è stata applicata al modulo di stima online dei parametri. Il valore dei parametri calcolati durante l'esecuzione del test in confronto con quelli simulati in

Simulink sono riportati in Figura 36. Da notare che l'andamento della capacità *C* è riportato in scala semi-logaritmica in modo da visualizzare tutte le variazioni, che in questo caso avvengono su un range molto ampio di valori.

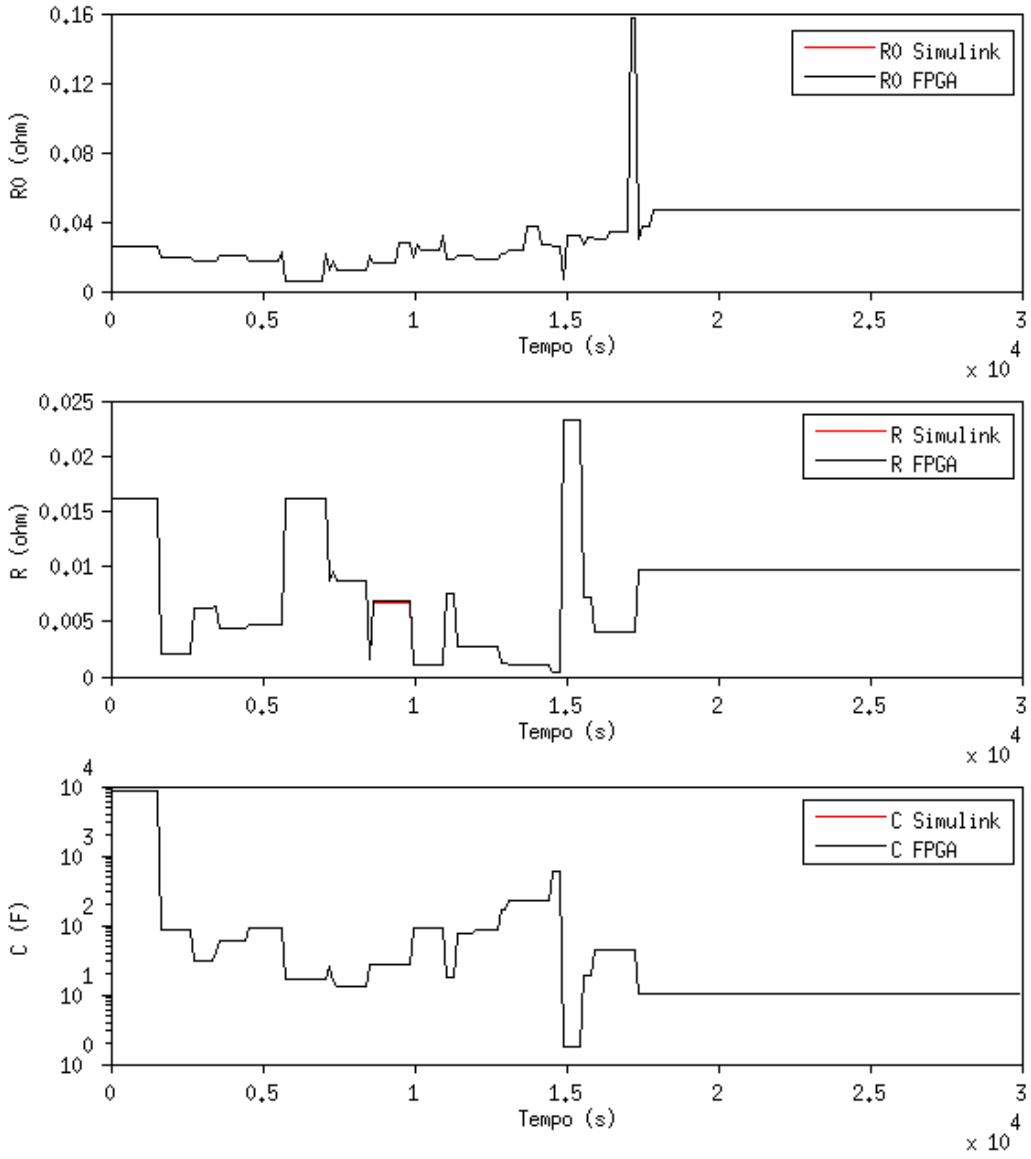

Figura 36: Confronto tra i parametri calcolati su FPGA e quelli simulati in ambiente Simulink.

Per ottenere questi risultati i campioni sono stati decimati con un fattore (*decfact*) pari a 10. La finestra temporale ha quindi una lunghezza pari a:

$$t_W = L \cdot decfact \cdot T = 30 \cdot 10 \cdot 0.1 = 30\,s$$

dove *L* è il numero di campioni nella finestra e *T* è il periodo di campionamento. La stima viene eseguita con un passo di 120 s.

Come si nota dalla figura, anche in questo caso i valori calcolati su FPGA sono coincidenti con quelli simulati in ambiente Simulink. Infatti, gli andamenti sono indistinguibili.

Entrambi i moduli hardware realizzati presentano risultati che sono identici a quelli ottenuti mediante simulazioni in ambiente Simulink. Risulta quindi evidente che sia il flusso di progetto utilizzato che l'implementazione realizzata sono validi.

È quindi possibile realizzare dei test reali sulla cella a disposizione al fine di testare anche la validità dell'algoritmo e non solo della sua implementazione.

5.7 Prestazioni

Da un'analisi effettuata durante l'esecuzione del test è stato possibile determinare le prestazioni del sistema.

Modulo	N_{exec}	T_{exec}
Stima del SOC	119	2.38 µs
Identificazione online dei parametri	6376	127.52 µs

Tabella 6: Tempi di esecuzione per una frequenza di clock di 50 MHz.

Nella Tabella 6 sono riportati i tempi di esecuzione (T_{exec}) e il numero di cicli di clock (N_{exec}) che i due moduli hardware realizzati impiegano ad effettuare il calcolo. Uno dei vantaggi più importanti derivanti dall'implementazione hardware dell'algoritmo è quello di poter effettuare la stima dello stato di più celle, utilizzando i moduli realizzati in time multiplexing. Il numero di celle (N_{cells}) che il sistema riesce a gestire dipende essenzialmente dal tempo di esecuzione del calcolo e dal periodo di campionamento degli ingressi (T_s):

$$N_{cells} = \frac{T_s}{T_{exec}} = \frac{T_s}{T_{clk} \cdot N_{exec}} = \frac{f_{clk}}{f_s \cdot N_{exec}}$$

dove f_s è la frequenza di campionamento degli ingressi e f_{clk} è la frequenza di clock del sistema.

Perché il sistema realizzato funzioni correttamente è necessario che i calcoli, per ogni cella, si concludano nel periodo di tempo che intercorre tra la ricezione di due campioni di tensione e di corrente consecutivi. Utilizzando i dati ottenuti, e ricordando che il periodo di campionamento utilizzato nel test è pari a 0.1 s, è possibile notare che si può stimare lo stato di circa 780 celle, cioè un numero tale da poter gestire l'intero pacco batteria di un veicolo elettrico o ibrido.

In realtà il sistema non è stato ottimizzato per ottenere il massimo numero di celle da gestire. Infatti, con determinate modifiche all'algoritmo è possibile incrementare il numero di celle gestibili raggiungendo un valore molto elevato.

Conclusioni

In questa tesi è stata realizzata un'implementazione a livello hardware su FPGA di un algoritmo di stima dello stato di carica di batterie al litio che utilizza un modello elettrico equivalente della cella. I parametri del modello dipendono dalle condizioni di utilizzo della batteria. Per questo motivo è stato implementato anche un algoritmo di identificazione online dei parametri. Le due parti dell'algoritmo vengono eseguite in parallelo al fine di ottenere una stima dello stato di carica il più accurata possibile.

L'algoritmo è stato prima descritto nel dettaglio, spiegando le tecniche e le metodologie utilizzate sia per quanto riguarda la stima del SOC che per l'identificazione online dei parametri.

Successivamente, sono stati riportati i passi implementativi per la realizzazione dell'algoritmo su FPGA. Per realizzare il sistema si è usato un flusso di progetto alternativo che fa uso del tool di sviluppo DSP Builder di Altera che permette di realizzare e simulare un modello dell'algoritmo in ambiente Simulink e di effettuare una sintesi hardware automatica.

Per concludere, si è effettuata una prova al fine di convalidare l'implementazione e il flusso di progetto adottato, utilizzando dati provenienti da misurazione eseguite in un test precedente. Il test in questione comprendeva l'applicazione di un profilo di corrente UDDS ad una batteria al litio NMC da 1.5 Ah, con lo scopo di studiarne il comportamento in un'applicazione reale come l'utilizzo in un veicolo elettrico. La prova è stata eseguita implementando il sistema su un FPGA Altera appartenente alla famiglia

Cyclone V, presente sulla scheda di sviluppo SoCKit. I risultati ottenuti sono del tutto soddisfacenti, in quanto il comportamento del sistema risulta del tutto coerente con le simulazioni effettuate in Simulink.

Sono state analizzate anche le prestazioni del sistema. I tempi di calcolo sono molto bassi e quindi implementare l'algoritmo descritto come acceleratore hardware e utilizzarlo in time multiplexing permette di poter stimare lo stato di un pacco batteria formato da un gran numero di celle, come ad esempio quelli che si trovano sui veicoli elettrici e ibridi.

Il sistema realizzato può essere implementato su FPGA relativamente a basso costo in quanto l'occupazione di risorse è di circa la metà di quelle presenti nel dispositivo a disposizione.

L'algoritmo implementato può essere ulteriormente migliorato sviluppando altri moduli hardware, come ad esempio un algoritmo per la stima dello stato di salute, che lavorano in parallelo ai due già implementati al fine ottenere una maggiore accuratezza nella stima del SOC.

Bibliografia

[1] X. Chen, W. Shen, Thanh Tu Vo, Z. Cao and A. Kapoor, "An overview of lithium-ion batteries for electric vehicles", IPEC, 2012 Conference on Power & Energy, 2012 , pp. 230-235.

[2] Andrew C. Baughman and Mehdi Ferdowsi, "Double-Tiered Switched-Capacitor Battery Charge Equalization Technique", IEEE Trans. on Industrial Electronics, vol. 55, no. 6, June 2008, pp. 2277-2285.

[3] S. Piller, M. Perrin and A. Jossen, "Methods for state-of-charge determination and their application", Journal of Power Sources, vol. 96, 2001, pp. 113-120.

[4] F. Baronti, W. Zamboni, N. Femia, R. Roncella, S. Rosi, R. Saletti, H. Raimi Eichi, and M.-Y. Chow, "Parameter Identification of Li-Po batteries in electric vehicles: a comparative study", 2013 IEEE International Symposium on Industrial Electronics, Sep. 2013, pp. 1-7.

[5] S. Rosi, "Stima dello stato di carica con identificazione on-line dei parametri di batterie al litio-polimero", Università di Pisa, Tesi di Laurea Specialistica, Dec. 2012.

[6] J. Cao, N. Schofield and A. Emadi, "Battery balancing methods: a comprehensive review", Vehicle Power and Propulsion Conference, 2008, pp. 1-6.

[7] Phillip Weicker , "A systems approach to Lithium-Ion Battery Management", Artech House, 2014.

[8] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 1. Background", Journalof Power Sources, vol. 134, 2004, pp. 252-261.

[9] C. Unterrieder, R. Priewasser, M. Agostinelli, S. Marsili and M. Huemer, "Comparative study and improvement of battery open-circuit voltage estimation methods", 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), Aug. 2012, pp. 1076-1079.

[10] BingXiang Sun, Lifang Wang, "The SOC Estimation of NIMH Battery Pack for HEV Based on BP Neural Network", International Workshop on Intelligent Systems and Applications 2009, ISA 2009, May 2009, pp. 1-4.

[11] M.A.C. Valdez, , J.A. Orozco Valera, M.J.O. Arteaga , "Estimating Soc in Lead-Acid Batteries Using Neural Networks in a Microcontroller-Based Charge-Controller", International Joint Conference on Neural Networks 2006, IJCNN '06 , 2006, pp. 2713-2719.

[12] Wei Jian, Xuehuan Jiang, Jinliang Zhang, Zhengtao Xiang, Yubing Jian, "Comparison of SOC Estimation Performance with Different Training Functions Using

Neural Network", 2012 UKSim 14th International Conference on Computer Modelling and Simulation (UKSim) , Mar. 2012, pp. 459-463.

[13] M. Chen, G. Rincon-Mora, "Accurate Electrical Battery Model Capable of Predicting Runtime and I-V Performance", IEEE Transactionon Energy Conversion, vol. 21, no. 2, Jun. 2006, pp. 504-511.

[14] Du Jiani, Wang Youyi, Wen Changyun, "Li-ion battery SOC estimation using particle filter based on an equivalent circuit model", 2013 10th IEEE International Conference on Control and Automation (ICCA) , Jun. 2013, pp. 580-585.

[15] T. Zahid, Guoqing Xu, Weimin Li, Lei Zhao, Kun Xu, "Performance analysis of particle filter for SOC estimation of LiFeP04 battery pack for electric vehicles", 2014 IEEE International Conference on Information and Automation (ICIA), Jul. 2014, pp. 1061-1065.

[16] M. Charkhgard, M. Farrokhi, "State-of-Charge Estimation for Lithium-Ion Batteries Using Neural Networks and EKF", IEEE Transactions on Industrial Electronics, vol. 57, no. 12, Dec. 2010, pp. 4178-4187.

[17] Liye Wang, Lifang Wang, Chenglin Liao, "Research on Improved EKF Algorithm Applied on Estimate EV Battery SOC", 2010 Asia-Pacific Power and Energy Engineering Conference (APPEEC), Mar. 2010, pp. 1-4.

[18] Tiezhou Wu, Xueguang Chen, Fangzhen Xia, Jianfeng Xiang, "Research on SOC Hybrid Estimation Algorithm of Power Battery Based on EKF", 2011 Asia-Pacific Power and Energy Engineering Conference (APPEEC), Mar. 2011, pp. 1-3.

[19] Dan Xu, Xiaoning Huo, Xin Bao, Changguang Yang, Hui Chen, Binggang Cao, "Improved EKF for SOC of the storage battery", 2013 IEEE International Conference on Mechatronics and Automation (ICMA), Aug. 2013, pp. 1497-1501.

[20] F. Codeca, S.M. Savaresi, G. Rizzoni, "On battery State of Charge estimation: A new mixed algorithm", IEEE International Conference on Control Applications, 2008 (CCA 2008), Sept 2008, pp. 102-107.

[21] T. Kim, Y. Wang, Z.Sahinoglu, T. Wada, S. Hara, W. Qiao, "Fast UD Factorization-Based RLS Online Parameter Identificationfor Model-Based Condition Monitoring of Lithium-ion Batteries", 2014 American Control Conference, June 2014.

[22] F. Mengali, "Implementazione su FPGA di un algoritmo per la stima dello Stato di Carica di batterie al Litio-Polimero", Università di Pisa, Tesi Magistrale.

[23] G. Golub, C. Van Loan, "Matrix Computations", The Johns Hopkins University Press, 4th edition, 2013.

[24] Zheng-Yu Huang, Pei-Yun Tsai, "High-throughput QR decomposition for MIMO detection in OFDM systems", Proceedings of 2010 IEEE International Symposium on

Circuits and Systems (ISCAS), May 30 - June 2 2010, pp. 1492-1495.

[25] Dai Haifeng, Wei Xuezhe, Sun Zechang, "State and Parameter Estimation of a HEV Li-ion Battery Pack Using Adaptive Kalman Filter with a New SOC-OCV Concept", 2009 International Conference on Measuring Technology and Mechatronics Automation, vol. 2, 11 - 12 April 2009, pp. 375-380.

[26] H. Rahimi-Eichi, F. Baronti, M.-Y. Chow, "Modeling and online parameter identification of Li-Polymer battery cells for SoC estimation", 2012 IEEE International Symposium on Industrial Electronics, May 2012, pp. 1336-1341.

[27] By the staff of Berkeley Design Technology, Inc., "An Independent Analysis of Floating-point DSP Design Flow and Performance on Altera 28-nm FPGAs", Oct. 2012.

[28] Altera, "DSP Builder Handbook, Volume 3: DSP Builder Advanced Blockset", 2013.

[29] Kokam Co., Ltd., "Cell Specification Data SLPB723870H4".

[30] [Online]. Available: http://www.epa.gov/nvfel/testing/dynamometer.htm, "Urban Dynamometer Driving Schedule (UDDS)".