POLITECNICO DI TORINO

Corso di Laurea Magistrale in
INGEGNERIA BIOMEDICA

Tesi di Laurea Magistrale

Modellazione comportamentale di circuiti digitali integrati con tecniche di apprendimento automatico

Relatori

Prof. IGOR SIMONE STIEVANO

Prof. RICCARDO TRINCHERO

Candidato

MARCO ATLANTE

23 Luglio 2024

Sommario

Nel panorama contemporaneo dello sviluppo di sistemi elettrici ed elettronici di nuova generazione, la disponibilità di modelli computazionali per i **circuiti digitali integrati** (IC) svolge un ruolo chiave. Sin dalle prime fasi di progetto, le simulazioni numeriche a livello di sistema risultano infatti uno strumento imprescindibile per la valutazione di potenziali criticità e la messa in atto di azioni correttive prima dello sviluppo finale e della commercializzazione dei prodotti. Un esempio tipico riguarda i sistemi digitali ad alte prestazioni come i più recenti dispositivi mobili (*smartphone*) o elaboratori (*notebook, workstation*).

In questo contesto, la complessità dei sistemi e la necessità di proteggere la proprietà intellettuale dei fornitori di circuiti integrati rendono impraticabili le descrizioni basate sulle caratteristiche fisiche del componente (es. *modelli transistor level*) che inoltre risulterebbero comunque inutilizzabili in un ambiente di simulazione in quanto eccessivamente onerosi, richiedendo tempi di simulazione lunghissimi.

La creazione di **modelli comportamentali**, generati a partire dalla osservazione del comportamento dei dispositivi, emerge naturalmente come una possibile soluzione, introducendo il vantaggio aggiuntivo di semplificare la procedura di stima dei parametri dei modelli e consentendo di ottenere caratterizzazioni compatte, accurate ed efficienti. I modelli ottenuti possono essere impiegati in simulazioni analogiche del comportamento elettrico di porzioni critiche di interconnessioni per valutare la robustezza del progetto mediante valutazioni di deterioramento e **integrità dei segnali** trasmessi e ricevuti.

Questa tesi parte da una analisi dello stato dell'arte che vede prevalentemente l'impiego di tecniche di modellazione basate su equivalenti circuitali o definite da strutture rigide di modello, fornendo successivamente una soluzione alternativa alla **modellazione comportamentale** di circuiti integrati mediante l'impiego di metodi a scatola nera. In specifico, ci si concentra su metodi recenti di **apprendimento automatico** e strutture di modello definite da **regressioni** del tipo Kernel Ridge Regression (KRR). Le strutture proposte offrono vantaggi in termini di generalità e capacità di modellazione del comportamento ricco delle dinamiche di questi dispositivi, superando molti del limiti noti di altri approcci quali quelli basati sulle reti neurali. Inoltre, è stato possibile progettare una procedura di stima

dei modelli basata sulla osservazione del comportamento dei dispositivi quando questi operano in condizioni di funzionamento normale quando connessi ad alcuni carichi rappresentativi.

I risultati ottenuti, basati sulla applicazione delle tecniche sviluppate a un dispositivo commerciale, dimostrano che i modelli KRR sono in grado di catturare accuratamente le dinamiche dei circuiti di interesse, consentendo previsioni affidabili delle loro prestazioni in condizioni variabili di carico, offrendo un valido supporto ai progettisti di sistemi integrati per generare modelli per la simulazione circuitale che potranno essere impiegati per migliorare la robustezza del progetto. La tesi offre inoltre un ottimo punto di partenza per la sperimentazione delle tecniche di modellazione comportamentale studiate in altri contesti applicativi di interesse ingegneristico dove è necessario disporre di modelli computazionali accurati ed efficienti.

Ringraziamenti

Desidero esprimere la mia più sincera gratitudine ai miei professori e relatori, il Prof. Igor S. Stievano e il Prof. Riccardo Trinchero, per il loro enorme supporto e la loro costante disponibilità durante questi mesi di lavoro.

La possibilità di collaborare con voi è stata un'opportunità che ha arricchito notevolmente il mio percorso accademico e personale. Grazie per avermi guidato con pazienza, per i preziosi consigli, e per avermi spronato a dare sempre il massimo. Il vostro sostegno è stato fondamentale per il completamento di questa tesi, e non posso che essere profondamente riconoscente per la fiducia che avete riposto in me e per l'entusiasmo con cui avete condiviso il vostro sapere.

La vostra dedizione e competenza sono state per me fonte di grande ispirazione. Vi ringrazio per aver reso questa esperienza così significativa e per aver contribuito in maniera decisiva alla mia crescita accademica e professionale.

Indice

Elenco delle figure VIII

1 Introduzione 1

2 Modellazione comportamentale di circuiti digitali integrati 3
2.1 Porte di ingresso e uscita (buffer I/O) 4
2.2 Modelli basati su IBIS 5
2.3 Approcci alternativi 9

3 Metodi di apprendimento automatico: evoluzione storica, regressioni e apprendimento 12
3.1 Machine Learning 14
3.1.1 Storia ed evoluzione del Machine Learning 16
3.1.2 Ciclo dell'Apprendimento Automatico: Fasi e Concetti Chiave 20
3.2 Tecniche di regressione 22
3.2.1 Regressione lineare e metodo dei minimi quadrati 24
3.2.2 Metodo dei minimi quadrati: regressione con funzioni di base 27
3.2.3 Ridge Regression 30
3.2.4 Kernel Ridge Regression 32

4 Stima di modelli dalle risposte alle porte dei dispositivi 35
4.1 Modellazione di tipo NARX 37
4.2 Carichi per l'addestramento e il Test 39

5 Analisi dei risultati ottenuti 46
5.1 Dispositivo reale scelto 47
5.2 Segnali di Training e di Test 47
5.3 Modello statico vs ricorrente 50
5.4 Validazione 51

6 Conclusioni e sviluppi futuri 55

A Differenziazione di matrici 57

Bibliografia 58

Elenco delle figure

2.1 Struttura tipica di un buffer di uscita di tipo single-ended con le relative variabili elettriche rilevanti (pannello superiore), illustrazione delle due principali strategie di macromodellazione (pannello inferiore). 4
2.2 Esempio di file IBIS. 7

4.1 Struttura di interconnessione tipica con i principali blocchi e le relative variabili elettriche di ingresso e uscita di un driver IC. . . . 36
4.2 Set-up utilizzato. 39
4.3 Test 1 - Linea di trasmissione #1 40
4.4 Netlist associata al Test 1 41
4.5 Test 2 - Carico RC 42
4.6 Test 3 - Carico resistivo in serie alla alimentazione 42
4.7 Test 4 - Carico resistivo 43
4.8 Test 5 - Linea di trasmissione #2 43

5.1 Risposte transitorie delle tensioni e delle correnti coinvolte osservate quando il dispositivo di test selezionato opera nelle 4 condizioni di carico usate per la stima. Test #1: linea di trasmissione con Z0=75 e TD=1.5e-9; #2: connessione shunt 50 Ohm // 10pF; #3: 50 Ohm in serie con la batteria di alimentazione VDD; #4: 150 Ohm. 48
5.2 Risposte transitorie delle tensioni e correnti coinvolte che sono associate al test #5: linea di trasmissione con Z0=50 e TD=2e-9. . 49
5.3 Validazione del modello sulla corrente di output di riferimento, $i_2(t)$, ottenuta sia dalla versione statica del modello che dalla versione ricorrente. 51
5.4 Validazione del modello sulla corrente di alimentazione, $i_3(t)$, ottenuta sia dalla versione statica del modello che dalla versione ricorrente. 52
5.5 Caratteristica statica del componente con le zone esplorate dal: test #1: linea di trasmissione con Z0=75 e TD=1.5e-9 e test #4: 150 Ohm. 53

5.6 Validazione del modello sulla caratteristica statica. 54

Capitolo 1

Introduzione

Negli ultimi anni, stiamo assistendo a un continuo e rapido incremento della velocità e della capacità di elaborazione della maggior parte dei sistemi digitali nel mondo dell'ingegneria elettrica ed elettronica, con un notevole aumento del cosiddetto **throughput dati** [1]. La tendenza sopra descritta si osserva in vari ambiti che vanno dal settore dell'ingegneria automobilistica o aerospaziale, alla elettronica di consumo e dei sistemi per la trasmissione e il processamento dei dati.

In questo contesto, giocano un ruolo chiave non solo i miglioramenti tecnologici dei circuiti integrati [2] ma anche le interconnessioni, cioè i mezzi fisici quali cablaggi o piste su circuito stampato utilizzati per trasferire dati tra dispositivi elettronici. In particolare, l'aumento della velocità di trasmissione dei dati, unitamente alla complessità dei circuiti digitali integrati, con integrazione spinta di logiche di processamento digitale nel silicio, richiedono simulazioni numeriche a livello di sistema delle porzioni critiche di interconnessione quali quelle tra i processori e le memorie. Ad esempio le simulazioni SI/PI (Signal Integrity/Power Integrity) sono essenziali per analizzare e garantire l'integrità del segnale, ossia la capacità di un sistema di mantenere la forma e l'intensità dei segnali elettrici durante la trasmissione la ricezione e l'elaborazione, necessari per il funzionamento corretto del dispositivo.

Con attenzione specifica al mondo dei dispositivi mobili come smartphone,

[1] I "throughput dati" si riferiscono alla quantità di dati che possono essere trasferiti da un sistema, un dispositivo o una rete in un determinato periodo di tempo. In altre parole, questo termine rappresenta la capacità di trasferimento dei dati ed è misurato in unità di dati per unità di tempo, come ad esempio bit al secondo (bps).

[2] I circuiti integrati sono fondamentali nell'elettronica moderna, poiché ospitano su piccoli chip di silicio un numero impressionante di transistor, dell'ordine del miliardi nelle piú recenti CPU (central Processing Unit).

tablet oltre a molti altri prodotti di consumo, questi oggetti rappresentano **ambienti complessi**, con dimensioni ridotte ma grande versatilità. Le velocità di trasmissione dei dati e i livelli di tensione sono adattati alle esigenze specifiche delle applicazioni, consentendo il trasferimento efficiente di grandi quantità di dati attraverso interconnessioni complesse. Questi collegamenti connettono le varie componenti chiave dei dispositivi mobili, come la memoria, il display, la batteria, i trasmettitori radio-frequenza e una serie di chip specializzati [1, 2].

Mentre gli utenti beneficiano delle molteplici funzionalità offerte da tali dispositivi, i progettisti devono affrontare problemi complessi legati alla progettazione di sistemi avanzati come schede a circuito stampato multistrato (PCB), sistemi in-package (SiP) o sistemi su chip (SoC). Questa progettazione richiede un'analisi approfondita della affidabilità del sistema, che può essere effettuata solo attraverso l'uso di strumenti di simulazione sofisticati e accurati che sfruttano descrizioni basate sulle leggi fisiche di funzionamento dei sistemi, quali ad esempio SPICE (Simulation Program with Integrated Circuit Emphasis). Tali strumenti devono consentire la predizione di fenomeni complessi legati alla distorsione dei segnali di tensione e di corrente nelle interconnessioni non ideali, effetti di diafonia e le fluttuazioni di alimentazione [3].

Tuttavia, l'utilizzo di simulazioni SPICE richiede la disponibilità di modelli accurati ed efficienti di tutte le porzioni del sistema, inclusi i circuiti digitali integrati e specificatamanete i buffer di ingresso e uscita (I/O, input/output), ovvero le porzioni di tali circuiti che sono presenti alle interfacce del dispositivo verso l'esterno e che connettono la logica interna (ovvero il mondo digitale) alle interconnessioni esterne. A causa della crescente complessità dei dispositivi, i modelli basati sulla fisica dei buffer (modelli transistor-level) risultano estremamente inefficienti, limitando fortemente il loro impiego in queste analisi. Al contempo, i modelli transistor-level, anche pensando di superare la problematica di onere computazionale, forniscono dettagli di progetto sulla logica costruttiva interna e i produttori di circuiti stampati cercano soluzioni alternative per non svelare informazioni proprietarie. Per superare questa sfida, è emersa nel corso degli ultimi trenta anni un'alternativa: l'utilizzo di **modelli surrogati**, detti anche **modelli comportamentali** o **macromodelli**.

Questi modelli sono progettati per garantire una rapida esecuzione computazionale e per simulare in modo preciso le risposte dei buffer, offrendo una caratterizzazione alle porte. Grazie a queste simulazioni più veloci, il flusso di progettazione può essere ottimizzato e la copertura della verifica a livello di sistema può essere estesa ad analisi più complesse.

All'interno di questo lavoro di ricerca ci si focalizza sulla modellazione comportamentale dei circuiti integrati digitali attraverso l'utilizzo di metodi di regressione basati sul kernel e tecniche di apprendimento automatico, offrendo una soluzione e uno strumento potente per la simulazione veloce di circuiti complessi.

Capitolo 2

Modellazione comportamentale di circuiti digitali integrati

Nel panorama attuale, la modellazione comportamentale dei circuiti digitali integrati riveste un ruolo cruciale nell'analisi e nella progettazione di sistemi complessi. Questo capitolo si focalizza sull'esplorazione di approcci e metodologie per la modellazione di circuiti digitali integrati, con particolare attenzione alle porte di ingresso e uscita (buffer I/O) e ai modelli basati sulla specifica **IBIS** (Input/Output Buffer Information Specification). I buffer I/O fungono da ponte tra il nucleo digitale del circuito integrato e il mondo esterno, garantendo un'efficiente trasmissione dei segnali elettrici. La loro corretta modellazione comportamentale è essenziale per garantire prestazioni affidabili e conformità agli standard di integrità del segnale e della alimentazione di tali circuiti.

Esploreremo l'utilizzo dei modelli basati su IBIS, una specifica di informazioni sui buffer I/O, che fornisce una rappresentazione standardizzata del comportamento elettrico di questi dispositivi mediante equivalenti circuitali. I modelli basati su IBIS offrono un approccio sistematico e accurato per la caratterizzazione dei buffer I/O, consentendo una simulazione efficiente e affidabile delle interconnessioni elettriche nei circuiti integrati. La complessità dei circuiti recenti ha peró richiesto continue modifiche e aggiustamenti della specifica, rendendo la procedura di raccolta delle informazioni decisamente elaborata e, in alcuni casi, offrendo modelli con limitata accuratezza.

Oltre agli approcci basati su IBIS, esamineremo anche approcci alternativi alla modellazione comportamentale dei circuiti digitali integrati, esplorando le sfide e le opportunità offerte da metodi innovativi, nati per semplificare le procedure di stima dei modelli e massimizzando la loro accuratezza.

2.1 Porte di ingresso e uscita (buffer I/O)

Il diagramma mostrato in Figura 2.1 illustra la tipica configurazione di un buffer di uscita (driver) di tipo *single-ended*, il quale collega la porzione interna, digitale, del circuito integrato (il *core*) alle interconnessioni esterne. Dal punto di vista generale, un macromodello di questo dispositivo richiede la definizione di una relazione dinamica non lineare che descrive il comportamento delle correnti $i(t)$ della porte di uscita e $i_{dd}(t)$ della porta di alimentazione in relazione alle tensioni corrispondenti $v(t)$ e $v_{dd}(t)$ e alla tensione di ingresso $v_{in}(t)$ forzata dalla logica interna. Queste relazioni possono essere applicate anche ai buffer di ingresso o a dispositivi di tipo differenziale. La sfida principale consiste nella scelta di relazioni dinamiche non lineari che siano precise, computazionalmente efficienti e facili da determinare e da implementare in un simulatore circuitale.

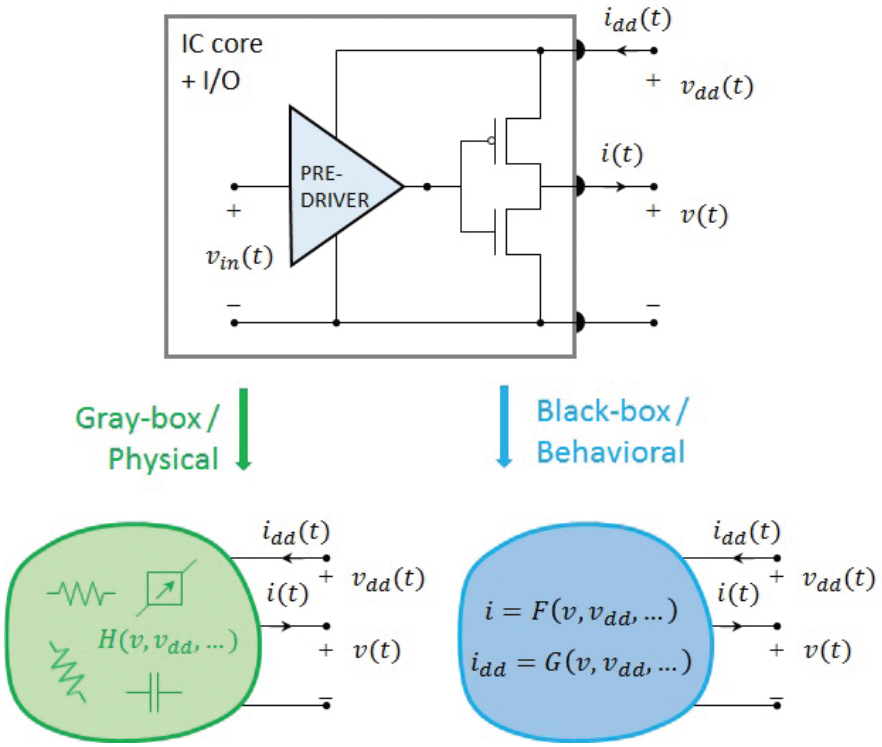

Figura 2.1: Struttura tipica di un buffer di uscita di tipo single-ended con le relative variabili elettriche rilevanti (pannello superiore), illustrazione delle due principali strategie di macromodellazione (pannello inferiore).

Fondamentalmente, emergono due strategie principali: un approccio "**bottom-up**", che si basa sulla progettazione di circuiti equivalenti semplici ispirati alla fisica interna del dispositivo, e un approccio "**top-down**", che cerca di identificare

un insieme ottimale di parametri per un surrogato matematico "**black-box**" ben scelto.

Anche se la classificazione completa può essere complicata, le soluzioni avanzate disponibili nella letteratura sembrano seguire uno di questi due approcci o combinare entrambi, nella ricerca di un miglioramento dell'accuratezza mantenendo procedure di stima dei parametri semplici e robuste [4]-[5].

2.2 Modelli basati su IBIS

La specifica **IBIS** [3], [4] è diventata il principale riferiemnto in ambito industriale per la caratterizzazione e la modellazione di buffer di circuiti integrati. Intel Corporation [1] ha iniziato lo sviluppo di IBIS nei primi anni '90, seguito dalla creazione dell'IBIS Open Forum (https://ibis.org/), supportato da aziende, utenti e accademici. Questo forum funge da centro di sviluppo e mantenimento della documentazione e degli strumenti IBIS, continuamente aggiornati attraverso il processo di raccolta di suggerimenti e miglioramenti tramite i **Buffer Issue Resolution Documents** (BIRD). Attualmente, IBIS è alla versione 7.3 ed è soggetto a costanti aggiornamenti.

IBIS si basa su modelli circuitali semplificati dei buffer di I/O tipici e fornisce linee guida dettagliate per l'estrazione delle caratteristiche rilevanti del dispositivo. Utilizzando una procedura di estrazione predefinita, IBIS permette di ottenere facilmente le caratteristiche statiche della corrente delle porte di uscita, la capacitá equivalente del die [2] di silicio, e altre informazioni cruciali. Questa specifica è ampiamente utilizzata per generare modelli di buffer e ha continuato a migliorare nel tempo, diventando uno standard ampiamente accettato. Tuttavia, in alcune applicazioni specifiche, si é resa necessaria una **maggiore accuratezza** rispetto a quella fornita da IBIS [6]-[5]. Di conseguenza, sono stati sviluppati approcci di modellazione alternativi, che mirano a considerare meglio l'effetto dinamico delle tensioni delle porte di I/O e di alimentazione sul comportamento del circuito, offrendo una descrizione più accurata della corrente delle porte di alimentazione.

Come accennato in precedenza, anziché fornire direttamente una descrizione del circuito equivalente, lo standard IBIS [4] suggerisce un elenco di caratteristiche che devono essere incluse in un file dati strutturato, come mostrato nella Figura 2.2. L'intestazione del file elenca gli elementi chiave che identificano il file IBIS e il dispositivo associato, che può essere un circuito integrato digitale con un numero

[1] Intel Corporation è una multinazionale statunitense leader nel settore dei semiconduttori e dei microprocessori, progetta e produce una vasta gamma di prodotti hardware tra cui processori per computer.

[2] Porzione di silicio su cui sono costruiti i circuiti integrati come i microprocessori.

variabile di pin di I/O. Si presume che ciascun pin del dispositivo sia collegato a un buffer di I/O corrispondente sul die di silicio attraverso un equivalente elettrico RLC raggruppato del package. La sezione dei dati del silicio è la parte più importante del file IBIS e contiene i set di dati che descrivono il comportamento di ogni porta IC [3]-[4]. Una descrizione completa delle parole chiave e delle caratteristiche di IBIS va oltre lo scopo di questa tesi.

La filosofia IBIS si basa sull'osservazione che tale dispositivo opera in uno **stato logico fisso** (alto o basso) o **transita tra stati logici** (da basso ad alto o da alto a basso). Il comportamento in uno stato logico fisso basso o alto è descritto da componenti di pull-up e pull-down, rappresentati dai transistor pMOS e nMOS (come mostrato in Figura 2.1), modellati come sorgenti di corrente controllate in tensione definite da tabelle di ricerca. Gli **effetti dinamici** dei componenti sono modellati con una approssimazione di dinamica lineare offerta da un condensatore.

La struttura risultante del modello IBIS può essere rappresentata matematicamente dall'equazione (2.1).

$$i(t) = k_{PU}(t) \cdot I_{PU}(v(t)) + k_{PD}(t) \cdot I_{PD}(v(t)) - C_{\text{COMP}} \frac{\delta v(t)}{\delta t} \tag{2.1}$$

dove

- $i(t)$: Corrente elettrica al tempo t.
- $k_{PU}(t)$ e $k_{PD}(t)$: Coefficienti di proporzione per la corrente di pull-up e pull-down rispettivamente.
- $I_{PU}(v(t))$ e $I_{PD}(v(t))$: Correnti di pull-up e pull-down, funzioni della tensione v(t). Queste correnti sono modellate in base alla tensione applicata al dispositivo e possono dipendere in modo non lineare da essa.
- C_{COMP}: Capacità del compensatore. Questo parametro rappresenta l'effetto dinamico del componente.

Ciascun componente dell'equazione (2.1) puó essere calcolato utilizzando le risposte del dispositivo a specifici stimoli e test bench durante il processo di modellazione. Ad esempio, le caratteristiche statiche vengono estratte dalle analisi **.DC sweep**, il componente Ccomp può essere calcolato con un'analisi **.AC** e le simulazioni transitorie vengono eseguite per ottenere coppie di tensione-tempo su carichi specifici.

Diversi moderni buffer di I/O ad alta velocità, specialmente quelli utilizzati nei collegamenti di memoria, sono particolarmente sensibili al rumore dell'alimentazione. Questo rumore può causare effetti dannosi come il jitter e le distorsioni del segnale, mettendo a rischio le prestazioni del sistema. Le fluttuazioni della tensione di alimentazione derivano dall'interazione dei parassiti della rete di distribuzione

```
                Header section
[IBIS ver]      6.1
[File name]     myIC.ibs
[Date]          01/04/2016
…
|
[Disclaimer]    …
[Component]     xyz
[Manufacturer]  …
[Package]
|             typ     min     max
R_pkg         0.02    0.01    0.03
L_pkg         1.5nH   1.0nH   2.0nH
C_pkg         0.5pF   0.4pF   0.6pF
|
                Pin-map section

|
[Pin]  signal_name   model_name …
|
1      A1            buffer1
2      A2            buffer2
…
100    GND           Ground
|
              Silicon data section

|
[Model]       buffer1
Model_type    Output
Polarity      Non-Inverting
Enable        Active-Low
|

     Data of buffer1, keywords: [C_comp],
   [Pulldown], [Pullup], [Rising Waveform],
             [Falling Waveform], …

|
[end]
```

Figura 2.2: Esempio di file IBIS.

di potenza con le correnti dinamiche di alimentazione associate agli eventi di commutazione del buffer di I/O. Pertanto, i modelli di buffer dovrebbero essere in grado di riprodurre accuratamente i profili di corrente di alimentazione e la dipendenza delle variazioni della tensione di alimentazione dalle caratteristiche di output.

Per affrontare questa sfida, lo standard IBIS ha aggiornato la sua struttura a partire dalla versione 5.0. Sono state introdotte due sezioni aggiuntive nei file IBIS: [$SSO_{PU/PD}$] e [Composite Current]. La sezione [$SSO_{PU/PD}$] contiene tabelle che considerano l'impatto delle variazioni statiche della tensione di alimentazione e di terra sulla resistenza delle caratteristiche di pull-up e pull-down. Le tabelle [Composite Current] raccolgono il profilo di corrente di alimentazione durante le transizioni di stato, fornendo informazioni sulle correnti dello stadio pre-driver e di altri componenti che attraversano il porto di alimentazione.

Nonostante questi miglioramenti, sono necessarie ulteriori ricerche per migliorare l'accuratezza dei modelli IBIS nella simulazione degli effetti della tensione di alimentazione sui tempi di commutazione e sulla corrente di alimentazione.

Inoltre, per supportare interconnessioni piú lunghe e velocità di dati piú elevate, sono state sviluppate diverse tecniche di equalizzazione per i trasmettitori e ricevitori SERDES ad alta velocità. Per affrontare gli effetti di tali tecniche, la specifica IBIS è stata estesa con l'**IBIS-AMI**, che consente di modellare l'intero canale di comunicazione dati utilizzando la teoria dei sistemi lineari tempo invarianti (LTI).

In conclusione, lo standard IBIS continua a evolversi per affrontare le sfide sempre più complesse nel modellare i buffer di I/O ad alta velocità, rimanendo un punto di riferimento importante nel campo del macro-modeling.

2.3 Approcci alternativi

Questa sezione riassume brevemente gli aspetti chiave di un insieme rappresentativo di approcci alternativi di macro-modellazione dei buffer di I/O disponibili in letteratura [6]-[5]. I metodi "**gray-box**" si basano su un certo livello di conoscenza della topologia fisica dei circuiti dei dispositivi; i metodi "**black-box**", invece, adottano tecniche puramente non lineari di identificazione input-output per risolvere i problemi di macro-modellazione. Tuttavia, la situazione pratica è piuttosto mista, e i ricercatori spesso cercano di combinare i vantaggi di entrambi gli approcci e alla fine si concentrano su ció che vedono come un compromesso accettabile.

Un primo tentativo naturale di fornire modelli di buffer migliorati é partire da un circuito basato su IBIS e introdurre modifiche adeguate. Nell'articolo [6], i circuiti equivalenti vengono modificati con l'inclusione di alcuni blocchi aggiuntivi che compensano possibili errori delle correnti di uscita previste e delle porte di alimentazione. I blocchi sopra menzionati sono implementati utilizzando elementi standard SPICE (ad esempio sorgenti di corrente controllate). Questa modifica migliora l'accuratezza del modello quando le tensioni di alimentazione oscillano.

In [7], [8], vengono presentati modelli potenziati che appartengono ancora a un approccio di circuito semplificato. I valori degli elementi del circuito sono definiti mediante tabelle o superfici multivariate costruite tramite toolbox standard e che tengono conto delle variazioni di processo-tensione-temperatura. Sono considerate sia topologie single-ended che differenziali.

In [9], [10] sono presentati modelli basati su IBIS di buffer di uscita, con una descrizione potenziata del comportamento dinamico non lineare dei dispositivi. Si pone particolare enfasi sulla sostituzione della capacità lineare di uscita C_{comp} con un elemento dinamico non lineare e sulla sua stima tramite una procedura ben definita. L'articolo [11] presta particolare attenzione al miglioramento del modello per tener conto degli effetti dello stadio pre-driver (ad esempio vedasi Figura 2.1) che potenzialmente introduce operazioni di overclocking del dispositivo. L'equivalente basato sulla fisica è supportato da una descrizione orientata a blocchi dello stadio pre-driver. Commenti simili valgono per [12] dove viene utilizzata la stessa filosofia per mimare la struttura dettagliata del circuito dei buffer.

Tutti i precedenti approcci condividono la stessa logica di modellazione, in cui viene adottato un circuito equivalente specializzato per accomodare tutte le caratteristiche non esplicitamente incluse in IBIS. In tutti questi approcci, la stima del modello si basa su una procedura di identificazione simile a quella delineata nella sezione precedente per i modelli IBIS nativi. Il principale svantaggio di questi approcci risiede nelle rigide e restrittive assunzioni a priori sulla struttura del modello. Con l'evoluzione e il miglioramento della tecnologia, alcune di queste ipotesi potrebbero non essere più valide e i metodi richiedono aggiornamenti e regolazioni continui.

Gli articoli [5] e [13] suggeriscono invece un metodo completamente a scatola nera, *black-box*. Viene utilizzato un modello basato su un'unica componente definita da una rete neurale per riprodurre le correnti delle porte del dispositivo come funzione di tutte le tensioni, inclusa esplicitamente la tensione di ingresso v_{in}. La prospettiva di un modello generale e completo é allettante. Tuttavia, sono richieste particolare attenzione e specifiche regolazioni per calcolare i parametri del modello per diversi dispositivi. Inoltre, sono necessari sforzi aggiuntivi per migliorare l'efficienza, la robustezza e la scalabilità del modello (ovvero, tenere conto di porte/pin aggiuntivi come uscite differenziali o porte di alimentazione, ecc.).

La **Macro-modellazione tramite Identificazione Parametrica delle Porte Logiche** (**Mpilog**) [14]-[15] é un'alternativa ben nota a IBIS che è cresciuta negli ultimi venti anni ed é riuscita a superare il confine tra accademia e industria. Mpilog cerca di generare modelli che uniscano i vantaggi sia degli equivalenti a circuito semplificato che delle rappresentazioni completamente black-box. Da un lato, mantiene il paradigma a due pezzi che anche IBIS usa, perché questa caratteristica è strumentale nel mantenere i modelli compatti e robusti. Gli eventi di commutazione sono modellati con semplici funzioni di pesatura, evitando la necessitá di algoritmi di identificazione ingombranti. Dall'altro lato, le rappresentazioni parametriche surrogate sostituiscono gli elementi semplici come le sorgenti controllate e i condensatori nella Figura 2.2. I parametri di queste strutture matematiche di uso generale possono essere facilmente stimati dall'osservazione delle risposte transitorie del buffer. Il problema di identificazione non lineare é piú semplice rispetto agli approcci puramente black-box, mentre il modello guadagna comunque un grado di libertà sufficientemente elevato per mimare il comportamento dinamico delle caratteristiche recenti dei dispositivi (un esempio tipico è la complessa risposta dinamica dei recenti driver differenziali con regolatori di tensione interni [15]).

In [16], viene utilizzato un approccio simile dove vengono utilizzate diverse funzioni di base per descrivere i modelli parametrici black-box (ad es., in termini di funzioni spline). I vantaggi di questo tipo di approccio ibrido sono molteplici: sia le tecnologie dei dispositivi single ended che differenziali sono gestite allo stesso modo, la stima del modello può essere fatta sia dalla simulazione numerica che dalle misurazioni. Inoltre, le operazioni di overclocking del dispositivo possono essere riprodotte senza affidarsi a conoscenze dettagliate della struttura interna. I modelli risultanti beneficiano di un'accuratezza superiore nella previsione dei modelli ad occhio e si comportano eccezionalmente bene nelle co-simulazioni SI/PI. Mpilog è uno strumento maturo basato su un framework ben definito che può essere facilmente utilizzato in un flusso di progettazione.

Come ultima osservazione, è importante notare un recente articolo [17] che tenta una rappresentazione black-box ispirata alla teoria dei circuiti. Gli autori utilizzano un modello non lineare ispirato da una struttura di tipo Thévenin in un tentativo di fornire una generalizzazione della maggior parte delle strutture a due

pezzi esistenti adottate in IBIS, Mpilog e altri [2]-[18]. L'articolo fornisce una prova di concetto avanzata, occupandosi solo di dispositivi single-ended e focalizzandosi principalmente sulla mimica accurata del comportamento dei circuiti di buffer durante gli eventi di commutazione, anche quando usati in scenari di overclocking. In questa ricerca, la metodologia esistente è stata estesa per tenere conto di ulteriori effetti come la tensione di alimentazione e la corrente di alimentazione assorbita dai buffer. Si tratta di modellare l'effetto della tensione di alimentazione su un segnale di uscita in modo semplice ma efficace. Questo viene fatto considerando la tensione di alimentazione $v_{dd}(t)$ come una fonte aggressiva indipendente che influenza la corrente di uscita dal terminale della porta dell'output. L'obiettivo è tener conto sia delle variazioni rapide dovute a problemi di compatibilità elettromagnetica, sia delle deviazioni più lente causate dall'alimentazione stessa. Per realizzare ciò, si presume che l'effetto di $v_{dd}(t)$ sulla tensione effettiva di soglia di commutazione sia trascurabile, concentrandosi invece sull'ampiezza della tensione. Un modello generale può essere costruito in modo efficiente utilizzando un'equazione che tiene conto sia del valore nominale di $v_{dd}(t)$, denotato come V_{DD}, sia di una tensione a circuito aperto, modellata come un generatore di tensione.

Da questa analisi dei molti approcci presenti in letteratura si puó osservare che la maggior parte di contributi si é sempre concentratata su strutture a due pezzi e su soluzioni ad-hoc, viste le limitazioni intrinseche di approcci a scatola nera con reti neurali. Tuttavia, i recenti sviluppi delle tecniche machine learning offrono strutture di modelli e procedure di stima caratterizzati da una maggiore flessibilità e capacità di adattamento, consentendo di catturare in modo più preciso le complesse relazioni tra la tensione di alimentazione e il comportamento del componente elettronico, semplificando la procedura di stima. Il lavoro di tesi si inserisce in questo contesto e si propone di offrire un contributo alla modellazione dei buffer esplorando tali tecniche e valutandone i benefici.

Capitolo 3

Metodi di apprendimento automatico: evoluzione storica, regressioni e apprendimento

L'**Intelligenza Artificiale**, definita anche come Artificial Intelligence, rappresenta un campo interdisciplinare che fonde informatica, filosofia e sociologia, mirando a creare programmi e sistemi tecnologici in grado di emulare le capacità umane di risolvere problemi e compiere azioni tipicamente associate alla mente umana [19]. Recentemente, l'intelligenza artificiale è stata identificata come la disciplina che si occupa dello sviluppo di macchine, sia hardware che software, capaci di operare autonomamente. La crescente attenzione su questo campo è alimentata dai progressi tecnologici, che includono sia avanzamenti nell'hardware, come sistemi di calcolo potenti e a basso consumo energetico, sia nell'analisi in tempo reale di enormi quantità di dati di vario tipo (noti come Big Data).

Se l'attributo "**artificiale**" è relativamente semplice da definire, il concetto di "**intelligenza**" è oggetto di ampio dibattito, con diverse interpretazioni a seconda della disciplina. John McCarthy, vincitore del Premio Turing nel 1971 e professore emerito di Informatica alla Stanford University, ha sottolineato nel suo lavoro del 2004 che l'intelligenza artificiale non si limita alla riproduzione dell'intelligenza umana, ma rappresenta la componente computazionale della capacità di raggiungere obiettivi [20]. Una definizione più recente e completa è stata proposta dall' AI HLEG [1] durante la Conferenza Europea del 2018, descrivendo l'intelligenza artificiale come

[1]AI HLEG o High-Level Expert Group on Artificial Intelligence rappresenta un gruppo di

sistemi, sia hardware che software, progettati dall'uomo per interpretare e elaborare informazioni dall'ambiente circostante al fine di raggiungere un obiettivo specifico [21].

E' possibile analizzare l'intelligeza artificiale da un punto di vista operativo, distinguendo tra approcci **top-down** e **bottom-up**. Il primo è basato su un sistema simbolico che identifica gli stati mentali all'interno di un sistema, risultando più comprensibile per gli esseri umani ma meno adatto a rappresentare la complessità e l'incertezza delle realtà mutevoli. Il secondo, invece, si concentra sulle architetture e sulle connessioni che costruiscono strutture e modalità di ragionamento più complesse, risultando più adatto a rappresentare i sistemi di intelligenza artificiale attualmente in uso, sebbene sia meno esplicito e di più difficile interpretazione [22] [23].

L'Intelligenza Artificiale è comunemente suddivisa in due categorie principali: l'Intelligenza Artificiale *forte* e quella *debole*. Queste categorie si distinguono per il livello di autonomia e capacità di ragionamento dei sistemi che le compongono.

L'**intelligenza artificiale debole** si riferisce a sistemi in grado di simulare alcune funzioni cognitive umane, ma che non raggiungono il loro livello di abilità. Ad esempio, i sistemi di intelligenza artificiale debole possono essere utili per svolgere compiti specifici come il riconoscimento di immagini o la traduzione di testi, ma richiedono ancora l'intervento umano per compiti più complessi. Il loro ragionamento si basa sull'analisi di casi precedentemente risolti e sull'apprendimento da essi, senza generare nuovi processi cognitivi autonomamente.

D'altro canto, l'**intelligenza artificiale forte** è caratterizzata da macchine con capacità cognitive indistinguibili da quelle umane. Questi sistemi si basano sulla tecnologia dei sistemi esperti, che mira a replicare le conoscenze e le abilità umane in un determinato settore. Ad esempio, un sistema di intelligenza artificiale forte potrebbe essere in grado di svolgere compiti complessi come il riconoscimento del linguaggio naturale e il pensiero astratto.

In questa distinzione, il **Machine Learning** è spesso associato all'intelligenza artificiale debole, poiché si basa sull'analisi di dati e pattern esistenti per fare previsioni o prendere decisioni, mentre il **Deep Learning** è visto come parte dell'intelligenza artificiale forte, in quanto utilizza reti neurali artificiali per emulare il processo di apprendimento umano, consentendo alle macchine di apprendere da grandi quantità di dati in modo autonomo.

esperti indipendenti di intelligenza artificiale costituito dall'Unione Europea nel giugno del 2018.

3.1 Machine Learning

Il **Machine Learning** rappresenta un ramo dell'informatica dedicato allo studio di sistemi e algoritmi che sono in grado di apprendere direttamente dai dati, senza richiedere una programmazione esplicita [24]. Il termine "*Machine Learning*" fu coniato nel 1959 da Arthur Lee Samuel, un pioniere dell'intelligenza artificiale [25], per descrivere la branca dell'informatica che utilizza algoritmi per interpretare e apprendere da dati non strutturati. Prima di approfondire questo concetto, è importante definire con precisione cosa si intende per "*apprendimento*".

Il professor Tom Michael Mitchell, direttore del dipartimento di Machine Learning presso la Carnegie Mellon University, lo descrive così: "*Un programma apprende da un'esperienza se, data una misura di prestazione associata a un problema, la sua prestazione nel risolvere il problema migliora con l'esperienza*" [26].

Gli algoritmi di Machine Learning elaborano esempi di dati forniti sotto forma di vettori di caratteristiche o "*feature*", valutando la loro capacità di risolvere problemi, tipicamente misurata in termini di performance. Nei problemi di classificazione, ad esempio, la performance può essere misurata in termini di accuratezza del modello, ossia la percentuale di predizioni corrette, o in base al tasso di errore, ovvero la percentuale di predizioni sbagliate.

I paradigmi dell'apprendimento automatico includono:

- **Apprendimento supervisionato**: si costruisce un modello a partire da dati di addestramento etichettati, in quanto l'obiettivo è elaborare una regola che associ input e output corrispondente;
- **Apprendimento non supervisionato**: si costruisce un modello a partire da dati di addestramento non etichettati o senza un corrispondente valore di output, in quanto l'obiettivo è identificare dei pattern negli input al fine di riprodurli o prevederli;
- **Apprendimento con rinforzo**: l'obiettivo è costruire un sistema che, attraverso le interazioni con l'ambiente, migliori le proprie performance.

La categorizzazione di cui sopra ha le sue radici nel lavoro di Samuel negli anni '50, egli basandosi sul modo in cui le macchine possono apprendere, distinse i diversi tipi di approcci principali [27]. Si noti che particolare attenzione viene posta sul primo approccio, poiché utilizzato nelle successive fasi e che, per dovizia di completezza, si accenna anche gli altri due.

L'**apprendimento supervisionato** implica l'uso di un set di addestramento in cui le etichette di classe sono note. L'obiettivo è predire l'output corretto in base alle caratteristiche di input, stimando la probabilità condizionale. Questo tipo di apprendimento mira a costruire modelli in grado di fare previsioni anche in condizioni di incertezza, migliorando iterativamente le previsioni grazie all'intervento di

un supervisore. Gli algoritmi di apprendimento supervisionato trovano applicazione in diversi settori, dall'ambito medico all'identificazione vocale, e sono in grado di formulare ipotesi induttive, ovvero generare soluzioni a partire da una serie di casi specifici. A seconda del tipo di output desiderato, si distinguono algoritmi di regressione, in cui l'output è *continuo*, e algoritmi di classificazione, in cui l'output è *discreto* e diviso in due o più classi [28].

L'**apprendimento non supervisionato**, invece, si basa su dati di input privi di output corrispondenti, l'obiettivo è individuare pattern o strutture nascoste nei dati forniti, senza l'intervento di un supervisore. Questo tipo di apprendimento è utile quando si hanno grandi quantità di dati non etichettati e si desidera comprendere il loro significato.

Il terzo tipo di apprendimento, definito anche come **Reinforcement Learning**, si basa sull'analisi dei feedback, premi e penalità. In questo caso, l'agente apprende attraverso l'interazione con l'ambiente, modificando il proprio comportamento per massimizzare i premi e ridurre le punizioni. Questo paradigma è tipico delle auto senza pilota, che grazie a un complesso sistema di sensori sono in grado di percorrere strade riconoscendo ostacoli e seguendo le indicazioni stradali.

3.1.1 Storia ed evoluzione del Machine Learning

Il concetto di **machine learning** venne originariamente coniato nel 1959 da Arthur Lee Samuel, un innovativo ricercatore americano nel campo dell'intelligenza artificiale. Tuttavia, la definizione più famosa di machine learning è attribuita a Tom Michael Mitchell, un altro eminente scienziato americano: "*Un programma apprende dall'esperienza E, in relazione a certe categorie di compiti T e con la misurazione delle performance P, se le sue performance nel compito T, misurate da P, migliorano con l'esperienza E*" [26].

In altre parole, attraverso l'esperienza, che può essere interpretata come il tempo nel contesto della mole di dati esaminati, la macchina ottimizza progressivamente le sue risposte alle sfide della realtà. Gli algoritmi di machine learning, basati su metodi matematico-computazionali, consentono al sistema informatico di adattare le proprie prestazioni in modo dinamico all'aumentare degli esempi da cui apprende.

La definizione di Mitchell è di notevole importanza poiché offre una formulazione operativa del concetto di apprendimento, piuttosto che basarsi su concetti cognitivi astratti. Con questa formulazione, Mitchell segue la linea di pensiero del celebre matematico britannico Alan Turing, il quale, nel suo articolo "*Computing Machinery and Intelligence*", affrontò la domanda "Le macchine possono pensare?" con un approccio differente, ponendo la controdomanda: "*Le macchine possono svolgere ciò che noi, esseri pensanti, possiamo fare?*". Turing è anche noto per aver introdotto nel 1950 un famoso test, proposto nello stesso articolo, che mira a valutare la capacità di un computer di emulare il comportamento umano. Questo test, ispirato al "**gioco dell'imitazione**", prevede che un esaminatore conduca una conversazione sia con un essere umano sia con una macchina, cercando di identificarne le rispettive identità. Una delle applicazioni più comuni del test di Turing è evidente nelle operazioni quotidiane online, come ad esempio durante le registrazioni sui siti web, dove vengono proposti **CAPTCHA** [2] che richiedono il riconoscimento di immagini anziché di numeri o testi. In sostanza, se un sistema informatico può sostituire con successo un essere umano, allora può essere considerato intelligente, in grado cioè di associare idee ed esprimerle. Nonostante il pensiero innovativo di Turing, il suo test ha ricevuto nel corso dei decenni diverse critiche, poiché non è considerato una prova sufficiente per valutare l'intelligenza effettiva di una macchina. A titolo di esempio, si può citare la cosiddetta "*stanza cinese*" [29], un esperimento mentale ideato da John Searle, che mette in discussione l'affidabilità del test di Turing, così come il programma **ELIZA** [3], che soddisfa i criteri del test nonostante sia

[2]Un CAPTCHA, che sta per "Completely Automated Public Turing-test-to-tell Computers and Human Apart", è un test utilizzato nell'informatica per distinguere tra utenti umani e bot tramite domande.

[3]ELIZA, creato nel 1966 dall'informatico tedesco Joseph Weizenbaum, è un bot per chat che

chiaramente un'entità priva di pensiero per sua stessa natura.

Il cambiamento di paradigma proposto da Turing fu fondamentale per gli studi iniziali sulla possibilità di sviluppare forme di intelligenza artificiale, che avrebbero preso forma solo pochi anni dopo. Sebbene la prima incarnazione del machine learning possa essere fatta risalire al 1943, quando McCulloch and Pitts introdussero il primo modello matematico che simulava una rete neurale artificiale [30], il termine "Artificial Intelligence" fu coniato e utilizzato per la prima volta nel 1956 durante la Dartmouth Conference. Questa conferenza, organizzata da Marvin Minsky (Harvard University), Claude Shannon (Bell Telephone Lab), Nathan Rochester (IBM) e John McCarthy (Dartmouth College), è stata cruciale nel delineare il campo dell'intelligenza artificiale. John McCarthy, in particolare, è considerato uno dei pionieri dell'intelligenza artificiale e la definì come "*la scienza e l'ingegneria della creazione di macchine intelligenti*". Durante la fine degli anni '50, ricercatori come Marvin Minsky, Arthur Samuel e Frank Rosenblatt contribuirono alla definizione dell'intelligenza artificiale attraverso metodi formali, approcci probabilistici e l'applicazione delle reti neurali, costituite in quel tempo da singoli percettroni [4] e modelli derivati dal modello lineare generalizzato della statistica.

Fu proprio in quegli anni che vennero avviate le prime simulazioni del cervello umano utilizzando le reti neurali (1957), e nacque il primo software di machine learning (1959). Arthur Samuel, lavorando per conto di IBM [5] sul primo calcolatore commercializzato dalla compagnia, l'IBM 701, sviluppò il primo gioco di dama "intelligente". In questo gioco, i computer miglioravano le proprie mosse interagendo con giocatori umani. Utilizzando un'analisi probabilistica delle posizioni possibili e delle mosse vincenti per ogni configurazione di gioco, il programma decideva le proprie mosse basandosi su una strategia di massimizzazione delle probabilità di vittoria, assumendo che l'avversario adottasse una strategia simile. Inoltre, grazie all'apprendimento mnemonico, il programma era in grado di memorizzare le posizioni già incontrate e di basarsi sull'esperienza passata per determinare le mosse più efficaci.

Il confronto del computer con giocatori professionisti e le sue numerose partite contro sé stesso, ripetute centinaia di volte, permisero al programma di evolversi

---

emula il ruolo di un terapeuta Rogersiano. Questo programma tende a rispondere riformulando le affermazioni fatte dal paziente stesso.

[4]Nel campo del machine learning, un percettrone è un tipo specifico di classificatore binario che associa un vettore di dati in ingresso x a un singolo valore scalare di output f(x). Questo concetto è stato introdotto da Frank Rosenblatt nel 1958.

[5]IBM, acronimo di International Business Machines Corporation, è una delle più grandi e storiche aziende nel settore dell'informatica e della tecnologia, nota per la produzione di hardware, software e servizi IT.

nel corso di oltre dieci anni fino a diventare in grado di sfidare giocatori di alto livello.

Nonostante i progressi iniziali, negli anni in cui si concentrava sullo studio dell'intelligenza artificiale, l'attenzione era principalmente rivolta agli approcci logici basati sulla conoscenza, creando una distanza significativa tra l'analisi dell'intelligenza artificiale e il suo sottoinsieme rappresentato dall'apprendimento automatico. Quest'ultimo, essendo necessariamente basato su un approccio probabilistico, si confrontava allora con diversi problemi applicativi, tra cui l'acquisizione e la rappresentazione di grandi quantità di dati che la scienza non era ancora in grado di gestire completamente. Inoltre, la ricerca sulle reti neurali subì un notevole rallentamento a seguito della pubblicazione degli studi di Minsky e Papert [31], in cui venivano evidenziate importanti limitazioni nell'utilizzo di percettroni e reti neurali.

Negli anni settanta e ottanta, l'attenzione nell'ambito dell'intelligenza artificiale si spostò completamente verso l'implementazione di **sistemi esperti** [6], trascurando completamente quelli basati su approcci statistici. La ricerca sull'apprendimento automatico venne portata avanti principalmente da ricercatori di altre discipline, al di fuori del campo dell'intelligenza artificiale, ottenendo inizialmente risultati significativi nel riconoscimento di pattern e nel recupero dell'informazione, e successivamente nell'approfondimento di algoritmi come la retropropagazione [7] e l'auto-organizzazione [8]. L'apprendimento automatico, sviluppatosi come disciplina separata dall'intelligenza artificiale tradizionale, ha iniziato a guadagnare terreno in modo significativo solo negli anni '90. L'obiettivo non era più quello di creare una forma di intelligenza artificiale simile a quella umana, ma piuttosto di risolvere problemi pratici specifici. Questo nuovo approccio, insieme all'interesse crescente per i modelli statistici basati sulla teoria della probabilità, ha segnato un forte progresso nella disciplina del machine learning. Fondamentale è stata la svolta dal metodo basato sulla conoscenza a quello basato sui dati.

Nel finale degli anni '90, IBM ottenne risultati eclatanti con **Deep Blue**, il primo computer a sconfiggere un campione del mondo di scacchi, il russo Garry Kasparov, in una partita con tempistiche regolamentari rispettate. La prima vittoria risale al

[6] Un sistema esperto è un programma di intelligenza artificiale che cerca di emulare le azioni di un esperto umano in un particolare campo di attività.

[7] La retropropagazione dell'errore è un algoritmo di apprendimento supervisionato ampiamente impiegato per l'implementazione delle reti neurali artificiali. Di solito, è combinato con un metodo di ottimizzazione come la discesa stocastica del gradiente.

[8] L'auto-organizzazione, conosciuta anche come ordine spontaneo nella teoria dei sistemi, è un processo mediante il quale un sistema si sviluppa attraverso forze ordinate e limitanti che provengono dalle componenti stesse che costituiscono l'ambiente oggetto di analisi. Questi sistemi presentano caratteristiche come complessità, autoreferenzialità, ridondanza e autonomia.

10 febbraio 1996, sebbene Kasparov riuscisse a vincere nel complesso per 4 partite a 2. Nell'incontro successivo, aggiornato, Deep Blue conquistò la rivincita l'11 maggio 1997, nonostante sospetti, in seguito confermati, riguardo a modifiche apportate al programma dai creatori tra una sfida e l'altra. Queste modifiche permisero al computer di sviluppare una intelligenza e creatività così profonde da risultare incomprensibili per l'avversario umano.

Nonostante le controversie, la vittoria di Deep Blue segnò una pietra miliare nell'evoluzione del machine learning, aprendo la strada a numerose applicazioni e implementazioni. Inoltre, contribuì significativamente ad aumentare l'interesse verso le scienze dell'intelligenza artificiale.

La nascita e l'espansione di Internet hanno segnato la definitiva affermazione degli studi nel campo dell'apprendimento automatico, grazie all'accelerazione esponenziale delle applicazioni digitali e alla facilità di reperimento e distribuzione dell'informazione, compresi i dati utilizzabili come campioni. Nei meccanismi dell'apprendimento automatico, i dati rivestono un ruolo cruciale, spesso più rilevante della stessa tecnologia, poiché costituiscono la vera fonte di conoscenza utilizzabile dal computer per apprendere. Questa è la ragione principale per cui il machine learning ha visto una diffusione così ampia solo negli ultimi quindici anni, nonostante le metodologie di base risalgano al secolo scorso.

3.1.2 Ciclo dell'Apprendimento Automatico: Fasi e Concetti Chiave

L'utilizzo delle tecniche di machine learning comporta il superamento della classica programmazione esplicita, in cui l'essere umano crea un modello basato su istruzioni del tipo "*if-then*", a favore di un metodo in cui la macchina è in grado di individuare autonomamente i pattern da seguire per ottenere il risultato desiderato. In generale, è possibile suddividere un processo di machine learning in tre fasi principali:

- **Apprendimento dei dati**: la macchina acquisisce e elabora i dati, che possono presentarsi in varie forme.
- **Valutazione dei dati**: il sistema informatico propone modelli statistici ipotetici per descrivere la realtà osservata.
- **Ottimizzazione dei modelli stimati e formulazione di una strategia di risposta/ azione**: in base ai feedback raccolti attraverso l'esperienza, il sistema migliora i modelli e sviluppa una strategia per le risposte o le azioni da intraprendere.

È all'interno di questa prima partizione che risulta importante introdurre la terminologia che si ritroverà nel corso del presente lavoro di ricerca. Iniziamo analizzando il concetto di dati, che costituiscono gli elementi con cui il sistema di apprendimento automatico interagisce. È importante notare che non tutti i dati sono uguali e non hanno la stessa importanza. Nella fase iniziale del processo di apprendimento, il sistema riceve un insieme di **dati destinati all'addestramento**, noto comunemente come **training dataset**.

Durante questa fase, il computer cerca di individuare relazioni tra i dati di input e quelli di output. Tali relazioni individuate rappresentano i parametri, o pesi, del modello stimato dal sistema. A volte, i dati di input noti anche come esempi sono accompagnati dai rispettivi dati di output, cioè i risultati attesi. In altre parole, il sistema riceve dati già "etichettati" o classificati, in cui è indicata una specifica realizzazione futura determinata dalle azioni intraprese.

Dopo aver ricevuto i training data, il sistema procede stimando le strutture ricorrenti e le regole logiche osservate negli esempi forniti, al fine di **determinare modelli e parametri** che descrivono la realtà. Questi pattern, o modelli, cercano di descrivere il processo di generazione dei dati (Data Generating Process, DGP) [9] del set di dati. È importante notare che il modello stimato dal computer non è necessariamente statico o determinato una volta per tutte, ma è in continua

[9]Il DGP descrive il processo mediante il quale ciascuna osservazione all'interno di un insieme di dati è stata generata, ovvero le relazioni tra i parametri che hanno portato a risultati specifici

evoluzione. Dopo il primo addestramento, la macchina inizia ad applicare una strategia di azioni che ritiene ottimale per raggiungere l'obiettivo prefissato. Questa sequenza di operazioni può risultare efficace o meno, e ciò viene valutato attraverso una funzione di valore che fornisce un **feedback** al computer, il quale può basare le sue mosse future su questo feedback.

Durante il processo di addestramento, il computer migliora progressivamente le sue mosse in risposta alle situazioni particolari affrontate, grazie ai feedback ricevuti. Al termine dell'analisi del dataset di addestramento, il sistema, sulla base dell'esperienza accumulata, svilupperà un'idea su come comportarsi di fronte a nuovi dati simili a quelli precedenti. Tuttavia, l'aspetto cruciale per gli esseri umani è valutare l'efficacia delle previsioni su informazioni future che potrebbero differire da quelle già assorbite dal sistema e ricevere previsioni future affidabili.

In questa prospettiva, entra in gioco un'ulteriore categoria di dati chiamata "**test data**", utilizzata appunto per testare i risultati prodotti dalla macchina dopo la fase di apprendimento sui training data. L'insieme di regole, formule, azioni e operazioni logiche viste permette di definire il concetto di algoritmo: sia le macchine che gli esseri umani, attraverso l'esperienza, creano regole generali che si traducono in modelli di apprendimento. Questi modelli, a loro volta, consentono di sviluppare algoritmi di apprendimento necessari per risolvere un determinato problema.

L'algoritmo fornisce al sistema le azioni o operazioni che può eseguire in un dato contesto. Se il sistema è in grado di eseguire tali operazioni, utilizzerà queste informazioni per compiti successivi. Nonostante gli studi sugli algoritmi di machine learning siano in costante evoluzione, la teoria dell'apprendimento non offre alcuna garanzia sull'effettiva performance degli algoritmi. La finitezza dei dati e la mancanza di conoscenza sull'evoluzione futura del modello rappresentano di per sé dei limiti. Per ottenere le migliori prestazioni possibili, la complessità dell'ipotesi induttiva deve essere adeguata a quella del modello sottostante il training dataset.

I problemi di **underfitting** e **overfitting** derivano proprio da questa disparità. In particolare, l'overfitting si verifica quando un modello statistico complesso si adatta troppo alle osservazioni del training set a causa di un numero eccessivo di parametri rispetto al numero di dati campionari disponibili; il contrario vale per l'underfitting.

In altre parole, l'overfitting si manifesta quando un modello addestrato su un training set non è in grado di generalizzare il pattern appreso su un insieme più ampio di dati: risulta molto preciso sui dati di addestramento, ma poco utile su dati sconosciuti. Al contrario, l'underfitting si verifica quando non ci sono abbastanza dati per stimare un numero adeguato di parametri e quindi il modello risulta poco aderente al reale processo generatore dei dati (DGP). Per evitare questi problemi, è importante eseguire l'addestramento utilizzando anche dati imprevisti.

In ogni caso, ad oggi, non esiste una soluzione univoca a problemi di questo tipo: ogni applicazione, ogni task da risolvere, richiede modelli e algoritmi diversi.

3.2 Tecniche di regressione

Le tecniche di regressione sono molto utilizzate in diversi ambiti dell'ingegneria. Ad esempio regressioni lineari vengono spesso adottate per la stima delle curve di taratura, mentre regressioni non lineari vengono usate per la costruzione di **modelli surrogati**, conosciuti anche come **metamodelli**, o per **la system identification**.

Nell'ambito ingegneristico, tali tecniche sono usate per sviluppare modelli matematici in forma chiusa in grado di imparare e di approssimare accuratamente il legame input-output legato ad un set di dati. Questi dati, noti come training set, sono generalmente generati tramite una serie di simulazioni o di misure mediante il **modello computazionale**, indicato come $M(\cdot)$. Quest'ultimo rappresenta la descrizione deterministica più accurata del comportamento effettivo del sistema a disposizione durante la fase di modellazione. Esso è in grado di fornire, per qualsiasi configurazione dei parametri di input (ad esempio parametri geometrici, parametri circuitali, definizione dei materiali, ecc..), una previsione dell'output di interesse. Tale modello può variare in termini di fedeltà e complessità, passando da una semplice approssimazione analitica in forma chiusa a una descrizione più sofisticata basata su principi fisici del sistema in esame.

Il training set è costituito da un insieme deterministico di coppie che raccolgono varie configurazioni dei parametri di input e i relativi valori di output. Tali campioni vengono utilizzati in fase di addestramento per la costruzione di un modello surrogato, mediante l'applicazione di tecniche di regressione o interpolazione.

Il **modello surrogato** $\widetilde{M}(\cdot)$ noto anche come **metamodello**, rappresenta un'approssimazione veloce ed efficiente del comportamento input-output del modello computazionale. È costruito utilizzando una quantità limitata di dati di addestramento e offre una valutazione rapida delle previsioni del modello. La precisione del modello surrogato viene tipicamente valutata confrontando le sue previsioni con i dati forniti da un dataset di test, composto da configurazioni deterministiche dei parametri di input e i relativi output calcolati tramite il modello computazionale [32] [33] [34].

Nell'ambito del Machine Learning, esistono svariati tipi di tecniche di regressioni supervisionate, ognuna delle quali presenta diversi punti di forza e caratteristiche in modo da proporre un ventaglio di soluzioni per le diverse applicazioni. Tra le varie tecniche disponibili nello stato dell'arte è importante citare regressioni ai minimi quadrati con funzioni di base, regressioni con kernel e regressioni basate su reti neurali. La principale differenza tra le tecniche citate è la struttura del modello. In quanto sia le regressioni con funzioni di base che quelle con i kernel si basano su modelli di tipo lineare (modelli in cui i coefficienti stimati dalla regressioni appaiono in modo lineare nella struttura del modello), mentre le regressioni basate su reti neurali hanno a che fare con strutture non lineari più complesse.

Nelle seguenti sezioni di questo capitolo, ci concentreremo sulle tecniche di

regressione ai minimi quadrati con funzioni di base e regressioni con kernel, con particolare attenzione alla regressione Ridge e Kernel Ridge [35]. Quest'ultime sono spesso preferibili ad approcci più flessibili come le regressioni basate su reti neurali, in quanto l'utilizzo di un modello con struttura lineare consente di semplificare la fase di training e allo stesso tempo di ridurre in modo marcato il numero di training samples da utilizzare. Quest'ultima caratteristica è fondamentale per la costruzione di modelli surrogati, in quanto minimizza il costo computazionale per la generazione del training set, aumentando l'efficacia e i vantaggi legati all'uso di modelli surrogati.

3.2.1 Regressione lineare e metodo dei minimi quadrati

La regressione lineare con il metodo dei minimi quadrati è il caso più semplice tra le possibili regressioni ai minimi quadri, in quanto consente di esamina la relazione media tra la variabile dipendente y (l'output) e la variabile indipendente x (l'input) tramite una retta, nota come **retta di regressione**. A tale scopo assumiamo che la relazione tra l'input e l'output del nostro modello computazionale è nota, ed è data dalla seguente retta:

$$y = M(x) + \epsilon \tag{3.1}$$
$$M(x) = \beta_0 + \beta_1 x \tag{3.2}$$

dove:

- y: variabile dipendente,
- x: variabile indipendente,
- ϵ: termine di errore,
- $M(x)$: funzione predittiva,
- β_0: intercetta,
- β_1: pendenza.

Durante l'addestramento del modello di regressione, l'obiettivo è trovare i migliori coefficienti che definiscono la relazione tra le variabili di input e l'output desiderato, minimizzando l'effetto del termine di rumore, partendo da un training set $\{x_i, y_i\}_{i=1}^{L}$ con L campioni, dove le coppie x_i e y_i sono i valori della variabile dipendente e indipendente calcolati con il modello computazione in (3.2).

La **minimizzazione dei quadrati dei residui** è un approccio comunemente utilizzato per adattare una linea di regressione ai dati osservati. I **residui** sono le **differenze tra i valori osservati e quelli previsti** dal modello per ciascun punto dei dati. Il **metodo dei minimi quadrati** (**OLS**) è un metodo specifico utilizzato per stimare i parametri del modello di regressione lineare. OLS cerca di trovare la migliore linea di regressione attraverso i dati **minimizzando la somma dei quadrati dei residui**, cioè cercando di trovare i coefficienti (pendenza e intercetta per la regressione lineare semplice) che minimizzano la somma dei quadrati delle differenze tra i valori osservati e quelli previsti. Una volta addestrato il modello utilizzando OLS, possiamo utilizzare i coefficienti stimati per fare previsioni su nuovi dati.

L'equazione per i valori stimati è formulata come:

$$\hat{y} = \hat{M}(x) \tag{3.3}$$
$$\hat{M}(x) = \hat{\beta}_0 + \hat{\beta}_1 x \tag{3.4}$$

dove:

- $\hat{y}$: variabile dipendente stimata,
- x: variabile indipendente,
- $\hat{f}(x)$: funzione predittiva stimata,
- $\hat{\beta}_0$: intercetta stimata,
- $\hat{\beta}_1$: pendenza stimata.

L'**errore residuo** (e) è la differenza tra i valori reali e quelli stimati rispetto alla variabile indipendente, ed è rappresentato dall'equazione:

$$e = y - \hat{y} = (\beta_0 + \beta_1 x + \epsilon) - (\hat{\beta}_0 + \hat{\beta}_1 x) = (\beta_0 - \hat{\beta}_0) + (\beta_1 - \hat{\beta}_1)x + \epsilon \tag{3.5}$$

Perciò, la somma dei residui al quadrato, detta anche Residual Sum of Squares (**RSS**), è calcolata come:

$$RSS = \sum_{i=1}^{L} e_i^2 = \sum_{i=1}^{L} (y_i - \hat{y}_i)^2 \tag{3.6}$$

L'**obiettivo principale** di questa tecnica è minimizzare questa differenza al fine di ottenere i coefficienti dell'intercetta (bias) e della pendenza:

$$\arg\min_{\beta_0,\beta_1} \sum_{i=1}^{L} (y_i - \hat{y}_i)^2 = \arg\min_{\beta_0,\beta_1} \sum_{i=1}^{L} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2 = \arg\min_{\beta_0,\beta_1} \sum_{i=1}^{L} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{3.7}$$

Per individuare i parametri interessati, si procede differenziando l'equazione appena trovata due volte, una volta per ogni parametro.

Iniziamo differenziando prima per $\hat{\beta}_0$, da cui si ottiene:

$$2 \cdot \sum_{i=1}^{L} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-1) = 0 \tag{3.8}$$

Svolgendo i passaggi successivi, si arriva a:

$$\hat{\beta}_0 = \frac{\sum_{i=1}^{L}(y_i - \hat{\beta}_1 x_i)}{L} \tag{3.9}$$

$$\hat{\beta}_0 = \sum_{i=1}^{L} \frac{y_i}{L} - \hat{\beta}_1 \sum_{i=1}^{L} \frac{x_i}{L} \tag{3.10}$$

e quindi:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \cdot \bar{x} \tag{3.11}$$

Dove $\bar{y}$ ed $\bar{x}$ rappresentano rispettivamente la media della variabile dipendente y e la variabile indipendente x. Allo stesso modo si procede con il secondo parametro $\hat{\beta}_1$, da cui si ottiene:

$$2 \cdot \sum_{i=1}^{L}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)(-x_i) = 0 \tag{3.12}$$

Svolgendo nuovamente tutti i passaggi, risulta:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{L} x_i y_i - L\bar{x}\bar{y}}{\sum_{i=1}^{L} x_i^2 - L\bar{x}^2} \tag{3.13}$$

In conclusione, attraverso il processo di minimizzazione dei quadrati dei residui, siamo stati in grado di ottenere le stime dei coefficienti $\hat{\beta}_0$ e $\hat{\beta}_1$, rappresentanti rispettivamente l'intercetta e la pendenza della retta di regressione. Questi coefficienti ci forniscono informazioni cruciali sulla relazione tra la variabile indipendente e la variabile dipendente nel modello di regressione lineare semplice, consentendoci di fare previsioni accurate su nuovi dati.

3.2.2 Metodo dei minimi quadrati: regressione con funzioni di base

La **regressione multipla** con funzioni di base è un'estensione della regressione lineare al caso in cui le relazioni tra più variabili dipendenti e la variabile indipendente è di tipo non lineare e non nota. Questo approccio è particolarmente utile quando ci sono diversi fattori che possono influenzare il risultato desiderato, come spesso accade in ambito scientifico.

A differenza della regressione lineare, in questo caso assumiamo che il modello computazione M, sia una generica mappa non lineare tra le variabili indipendenti $\boldsymbol{x}$, dove $\boldsymbol{x} \in \mathbb{R}^p$, e la variabile dipendente $y \in \mathbb{R}$, tale che:

$$y = M(\boldsymbol{x}). \tag{3.14}$$

e

$$M : \mathbb{R}^p \to \mathbb{R}. \tag{3.15}$$

Non avendo informazioni sulla struttura di M, il modello usato all'interno della regressione può essere scritto come combinazione lineare di un set D di funzioni di base ϕ_d:

$$\hat{y} = \hat{M}(\boldsymbol{x}) = \hat{\beta}_1\phi_1(\boldsymbol{x}) + \hat{\beta}_2\phi_2(\boldsymbol{x}) + \ldots + \hat{\beta}_D\phi_D(\boldsymbol{x}), \tag{3.16}$$

che può essere riscritto in forma compatta come:

$$\hat{y} = \hat{M}(\boldsymbol{x}) = \sum_{d=1}^{D} \hat{\beta}_d\phi_d(\boldsymbol{x}). \tag{3.17}$$

Partendo da un training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{L}$ con L campioni, dove le coppie $\boldsymbol{x}_i$ e y_i sono i valori della variabile indipendente e dipendente calcolati con il modello computazionale. L'obiettivo è determinare i coefficienti $\hat{\beta}_d$ in (3.17) che minimizzano l'errore residuo al quadrato tra i valori osservati della variabile dipendente y e i valori predetti dal modello di regressione $\hat{y}$:

$$RSS = \sum_{i=1}^{L} e_i^2 = \boldsymbol{e}^T\boldsymbol{e} \tag{3.18}$$

dove

$$\boldsymbol{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_L \end{pmatrix} = \begin{pmatrix} y_1 - \hat{y_1} \\ y_2 - \hat{y_2} \\ \vdots \\ y_L - \hat{y_L} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{pmatrix} - \begin{pmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_L} \end{pmatrix} = \boldsymbol{y} - \hat{\boldsymbol{y}}$$

Sviluppando l'equazione (3.18) con la notazione matriciale otteniamo:

$$\begin{aligned} RSS = \boldsymbol{e}^T\boldsymbol{e} = & \\ &= (\boldsymbol{y} - \hat{\boldsymbol{y}})^T(\boldsymbol{y} - \hat{\boldsymbol{y}}) = \\ &= (\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{\beta}})^T(\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{\beta}}) = \\ &= (\boldsymbol{y}^T - \hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T)(\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{\beta}}) = \\ &= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{\Phi}\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{y} + \hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\hat{\boldsymbol{\beta}}, \end{aligned} \tag{3.19}$$

dove

- $\boldsymbol{y} = [y_1, \ldots, y_L]^T \in \mathbb{R}^L$:è il vettore delle risposte osservate nel dataset di addestramento, i valori veri della variabile dipendente.
- $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_L]^T \in \mathbb{R}^L$ è il vettore delle risposte predette dal modello di regressione lineare.
- $\boldsymbol{\Phi} = [\boldsymbol{\Phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\Phi}(\boldsymbol{x}_L)]$ è la matrice delle funzioni di base, composta da valori ottenuti valutando le funzioni di base sui valori di $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L\}$ nel dataset di addestramento.
- $\boldsymbol{\Phi}^T$ è la trasposta della matrice delle funzioni di base.
- $\hat{\boldsymbol{\beta}}$ è il vettore dei coefficienti del modello stimato durante il processo di addestramento.
- $\hat{\boldsymbol{\beta}}^T$ è il trasposto del vettore dei coefficienti stimati.

Utilizzando la differenziazione delle matrici (Appendice A) andiamo a derivare l'equazione appena trovata rispetto a $\hat{\boldsymbol{\beta}}$ e poniamo tutto uguale a 0:

$$\begin{aligned} \frac{\delta(\text{RSS})}{\delta\hat{\boldsymbol{\beta}}} =& \frac{\delta(\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{\Phi}\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}\boldsymbol{\Phi}^T\boldsymbol{y} + \hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\hat{\boldsymbol{\beta}})}{\delta\hat{\boldsymbol{\beta}}} = 0 \\ & \frac{\delta(\boldsymbol{y}^T\boldsymbol{y})}{\delta\hat{\boldsymbol{\beta}}} - \frac{\delta(\boldsymbol{y}^T\boldsymbol{x}\hat{\boldsymbol{\beta}})}{\delta\hat{\boldsymbol{\beta}}} - \frac{\delta(\hat{\boldsymbol{\beta}}\boldsymbol{\Phi}^T\boldsymbol{y})}{\delta\hat{\boldsymbol{\beta}}} + \frac{\delta(\hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\hat{\boldsymbol{\beta}})}{\delta\hat{\boldsymbol{\beta}}} = 0 \\ & 0 - \boldsymbol{y}^T\boldsymbol{\Phi} - (\boldsymbol{\Phi}^T\boldsymbol{y})^T + 2\hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi} = 0 \\ & 0 - \boldsymbol{y}^T\boldsymbol{\Phi} - \boldsymbol{y}^T\boldsymbol{\Phi} + 2\hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi} = 0 \\ & 2\hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{x} = 2\boldsymbol{y}^T\boldsymbol{\Phi} \\ & \hat{\boldsymbol{\beta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{y}^T\boldsymbol{\Phi} \\ & \hat{\boldsymbol{\beta}}^T = \boldsymbol{y}^T\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1} \end{aligned} \tag{3.20}$$

Una volta completata la procedura di differenziazione e risolta l'equazione, otteniamo l'equazione che ci permette di calcolare $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}. \tag{3.21}$$

E' importante notare che la formulazione sviluppa in precedenza è equivalente alla minimizzazione della seguente funzione di costo:

$$L_{OLS}(\hat{\boldsymbol{\beta}}) = \sum_{i=1}^{L}(y_i - \hat{\boldsymbol{\beta}}\boldsymbol{\Phi}(\boldsymbol{x}_i))^2. \tag{3.22}$$

Una delle principali limitazioni di questa tecnica è l'**overfitting**, che si verifica quando il modello si adatta troppo al training set e perde la capacità di generalizzare su nuovi dati. Questo può accadere soprattutto quando il modello è troppo complesso rispetto alla complessità intrinseca dei dati. Inoltre, la soluzione in (3.21) assume che la matrice data dal prodotto $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})$ sia invertibile e ben condizionata. Tuttavia, in alcuni casi questa condizione potrebbe non essere soddisfatta, portando a problemi di stabilità nella stima dei coefficienti.

3.2.3 Ridge Regression

Per mitigare le limitazioni legate all'overfitting presenti nella regressione ai minimi quadrati, è possibile ricorrere a tecniche di regolarizzazione, come quelle adottate dalla **Ridge Regression** e dalla **LASSO Regression**. Il regolarizzatore aiuta a controllare l'overfitting e migliorare l'accuratezza nella stima dei coefficienti. Tra le due tecniche sopracitate, la Ridge regression ha il vantaggio di consentire la stima dei coefficienti della regressione in forma chiusa, mentre la LASSO regression usa un algoritmo numerico dedicato rendendo la fase di training più complicata.

Come nel caso della regressione ai minimi quadrati con funzioni di base, anche il modello usato dalla ridge può essere scritto come combinazione lineare di un set di funzioni di base ϕ_d:

$$\hat{\boldsymbol{y}} = \hat{M}(\boldsymbol{x}) = \sum_{d=1}^{D} \hat{\beta}_d \phi_d(\boldsymbol{x}). \tag{3.23}$$

Il modello in (3.23), può essere stimato minimizzando la seguente funzione costo:

$$\begin{aligned} L_{ridge}(\hat{\boldsymbol{\beta}}) &= \sum_{i=1}^{L} (y_i - \hat{\boldsymbol{\beta}}\boldsymbol{\Phi}(\boldsymbol{x}_i))^2 + \lambda \sum_{j=1}^{D} \hat{\beta}_j{}^2 \\ &= \sum_{i=1}^{L} (y_i - \hat{\boldsymbol{\beta}}\boldsymbol{\Phi}(\boldsymbol{x}_i))^2 + \lambda \left\| \hat{\boldsymbol{\beta}} \right\|^2 \end{aligned} \tag{3.24}$$

dove alla somma dei quadrati dei residui viene aggiunta una penalità tramite il termine $\lambda \left\| \hat{\boldsymbol{\beta}} \right\|^2 = \lambda \sum_{j=1}^{D} \hat{\beta}_j^2$ con $\lambda \geq 0$. Questo termine di penalità, noto come **penalità di ritiro** o **regolarizzatore**, aiuta a regolare l'effetto delle variabili predittive sul modello.

Il problema di minimizzazione descritto sopra può essere interpretato nella seguente formulazione matriciale:

$$\begin{aligned} L_{ridge}(\hat{\boldsymbol{\beta}}) &= \sum_{i=1}^{L} (y_i - \hat{\boldsymbol{\beta}}\boldsymbol{\Phi}(\boldsymbol{x}_i))^2 + \lambda \left\| \hat{\boldsymbol{\beta}} \right\|^2 \\ &= (\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{\beta}})^T (\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{\beta}}) + \lambda \hat{\boldsymbol{\beta}}^T \hat{\boldsymbol{\beta}} \\ &= \boldsymbol{y}^T \boldsymbol{y} - \boldsymbol{y}^T \boldsymbol{\Phi} \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}^T \boldsymbol{\Phi}^T \boldsymbol{y} + \hat{\boldsymbol{\beta}}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \hat{\boldsymbol{\beta}} + \lambda \hat{\boldsymbol{\beta}}^T \hat{\boldsymbol{\beta}} \end{aligned} \tag{3.25}$$

Anche in questo caso, ponendo a zero le derivate parziali di (3.25) calcolate rispetto a $\hat{\boldsymbol{\beta}}$, otteniamo:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}. \tag{3.26}$$

Dalla soluzione nella precedente equazione è facile notare che quando $\lambda = 0$, la penalità di ritiro non ha effetto e la regressione ridge produce stime dei coefficienti identiche a quelle della regressione ai minimi quadrati. Tuttavia, aumentando il valore di λ, la penalità di contrazione diventa più significativa aumentando il bias del modello e riducendone la varianza, e quindi contrastando l'overfitting. Tuttavia, è importante notare che un aumento eccessivo di λ può portare a una sottostima significativa dei coefficienti e ad un aumento della distorsione.

La Ridge Regression, pur offrendo numerosi vantaggi, presenta anche alcuni svantaggi che è importante considerare. Il problema principale delle regressioni basate sulla combinazione lineare delle funzioni di base è **il curse of dimensionality**, ovvero una riduzione dell'efficienza del modello all'aumentare della sua complessità. Infatti, come nel caso della regressione ai minimi quadrati, il numero di coefficienti $\hat{\beta}_j$ da stimare è fissato dal numero di funzioni di base usato dal modello, che ovviamente cresce in modo esponenziale con la complessità e con il numero di variabili.

La costruzione di modelli con un numero non trascurabile di funzioni di base (ad esempio sopra il migliaio) ha effetto sia sui tempi di calcolo, sia sul numero di training sample da usare. In quanto, nella formulazione considerata, una buona stima dei coefficienti del modello richiede un numero di training sample uguale o di solito superiore al numero di funzioni di base. Questo vuol dire che la costruzione di un modello accurato potrebbe richiedere la disponibilità di un numero enorme di dati nel training set. Inoltre è importante sottolineare che la scelta del parametro di regolarizzazione λ non può essere effettuata sul training set, ma richiede tecniche di ottimizzazione o metodi di validazione incrociata. Una scelta sbagliata di λ può portare a un modello sovra- o sotto-regolarizzato, compromettendo le prestazioni predittive.

3.2.4 Kernel Ridge Regression

La **Kernel Ridge Regression** (**KRR**) si presenta come una valida alternativa alla ridge regression in grado di limitare gli effetti della curse of diemensionality. La KRR sfrutta i vantaggi relativi derivati dall'impiego del kernel, legati al kernel trick. La trasformazione ottenuta mediante l'uso dei kernel è cruciale perché consente di eseguire la regressione in un nuovo spazio dove la relazione tra le variabili predittive e la variabile dipendente può essere modellata in modo più flessibile ed accurato. In sostanza, la KRR migliora la capacità del modello di adattarsi a strutture più complesse nei dati.

La KRR si basa sulla seguente interpretazione della soluzione proposta dalla Ridge regression (3.26), ovvero:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T y = \boldsymbol{\Phi}(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}, \tag{3.27}$$

dove questa volta nella seconda formulazione, anche detta formulazione nello spazio duale, la matrice da invertire ha dimensione $L \times L$ dove L è il numero di training samples.

Perciò, i coefficienti $\hat{\boldsymbol{\beta}}$ possono essere scritti come una combinazione lineare delle funzioni di basi pesata con dei nuovi coefficienti $\boldsymbol{\alpha}$:

$$\hat{\boldsymbol{\beta}}^* = \sum_{i=1}^{D} \alpha_i \boldsymbol{\Phi}(\boldsymbol{x}_i) \tag{3.28}$$

dove i coefficienti $\boldsymbol{\alpha}$ sono ottenuti come:

$$\boldsymbol{\alpha} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{y} \tag{3.29}$$

A questo punto possiamo definire la funzione kernel a partire dal prodotto di matrici $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$:

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{K} =$$

$$= \begin{pmatrix} \boldsymbol{\Phi}(\boldsymbol{x}_1)^T\boldsymbol{\Phi}(\boldsymbol{x}_1) & \boldsymbol{\Phi}(\boldsymbol{x}_1)^T\boldsymbol{\Phi}(\boldsymbol{x}_2) & \dots & \boldsymbol{\Phi}(\boldsymbol{x}_1)^T\boldsymbol{\Phi}(\boldsymbol{x}_L) \\ \boldsymbol{\Phi}(\boldsymbol{x}_2)^T\boldsymbol{\Phi}(\boldsymbol{x}_1) & \boldsymbol{\Phi}(\boldsymbol{x}_2)^T\boldsymbol{\Phi}(\boldsymbol{x}_2) & \dots & \boldsymbol{\Phi}(\boldsymbol{x}_2)^T\boldsymbol{\Phi}(\boldsymbol{x}_L) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Phi}(\boldsymbol{x}_L)^T\boldsymbol{\Phi}(\boldsymbol{x}_1) & \boldsymbol{\Phi}(\boldsymbol{x}_L)^T\boldsymbol{\Phi}(\boldsymbol{x}_2) & \dots & \boldsymbol{\Phi}(\boldsymbol{x}_L)^T\boldsymbol{\Phi}(\boldsymbol{x}_L) \end{pmatrix} =$$

$$= \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_L) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_2, \boldsymbol{x}_L) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ k(\boldsymbol{x}_L, \boldsymbol{x}_1) & k(\boldsymbol{x}_L, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_L, \boldsymbol{x}_L) \end{pmatrix}$$

dove per ogni combinazione delle variabili di input $\boldsymbol{x}$ e $\boldsymbol{x}'$, la funzione kernel è definita come il prodotto interno tra le funzioni di base:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \phi_d(\boldsymbol{x})\phi_d(\boldsymbol{x}') = \boldsymbol{\Phi}(\boldsymbol{x})^T\boldsymbol{\Phi}(\boldsymbol{x}'). \tag{3.30}$$

Usando la definizione della funzione k nell'equazioni precedenti, i coeffienti $\boldsymbol{\alpha}$ possono essere calcolati come:

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}. \tag{3.31}$$

dove la matrice quadrata $\boldsymbol{K}$ è il gramiano del kernel ($[K]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$) le cui dimensioni dipendono dal numero di training sample L .

Un vantaggio fondamentale della KRR è che non è più necessario conoscere la funzioni di base usate nella matrice $\boldsymbol{\Phi}(\cdot)$ in modo esplicito, perchè tali informazioni sono incorporate nella funzione kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

La KRR è quindi formulata come:

$$\widetilde{M}(\boldsymbol{x}) = \sum_{l=1}^{L} k(\boldsymbol{x}, \boldsymbol{x}_l)\alpha_l \tag{3.32}$$

Ecco alcuni esempi comuni di funzioni kernel:

- Kernel lineare: $k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{x}_m^T \cdot \boldsymbol{x}_n$
- Kernel Polinomiale: $k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^T \cdot \boldsymbol{x}_n + r)^d$
- Kernel Gaussiano: $k(\boldsymbol{x}_m, \boldsymbol{x}_n) = e^{-\frac{\|\mathbf{x_m} - \mathbf{x_n}\|^2}{2 \cdot \sigma^2}}$

La kernel ridge regression rappresenta una potente combinazione di regressione ridge e trucco del kernel, offrendo un approccio flessibile ed efficace per affrontare problemi di regressione con dati complessi e non lineari. Infatti la complessità del problema di regressione in (3.32) in termini del numero di coefficienti da stimare è indipendente dal numero di funzioni di base, ma è fissato dal numero di training sample.

Capitolo 4

Stima di modelli dalle risposte alle porte dei dispositivi

Nel panorama delle simulazioni circuitali per la signal and power integrity (SI/PI), l'adozione diffusa di modelli basati su strutture come la classe IBIS [4] o Mpilog [14] [15] ha indubbiamente portato a livelli di accuratezza ed efficienza notevoli. Tuttavia, una sfida persistente rimane nella procedura di stima di tali modelli, la cui generazione richiede infatti configurazioni specifiche e talvolta procedure complesse per l'eccitazione dei dispositivi e l'osservazione delle loro risposte, che vengono successivamente impiegate per il calcolo dei parametri o degli elementi circuitali delle strutture considerate.

Uno degli obiettivi principali di questo studio è il tentativo di semplificare la procedura di stima mediante l'utilizzo di regressioni e modelli di machine learning, come descritto nel capitolo 3. L'ideale sarebbe poter osservare la risposta transitoria delle tensioni e delle correnti alle porte di un circuito integrato digitale durante condizioni di funzionamento normali, quando è inserito in uno schema di progetto tipico, ad esempio per l'analisi della comunicazione tra una CPU e una memoria (vedasi Figura 4.1).

In particolare, si mira a trovare una relazione approssimata che descriva la caratteristica multiporta del driver, la cui forma implicita è definita tramite la funzione $g(\cdot)$ che rappresenta una mappa generica e dinamica non lineare:

$$g(\mathbf{u}(t), y(t), t, d/dt, d^2/dt^2, \ldots) = 0 \tag{4.1}$$

dove $\mathbf{u}(t) = [v_i(t), v_o(t)]^T$ e $y(t) = i_o(t)$. La rappresentazione potrá includere più variabili quali la tensione di alimentazione e la corrispondente corrente di

alimentazione.

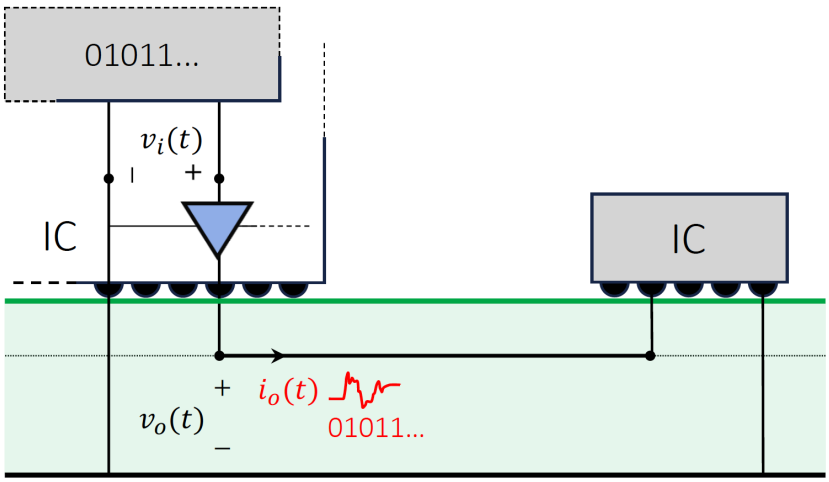

Figura 4.1: Struttura di interconnessione tipica con i principali blocchi e le relative variabili elettriche di ingresso e uscita di un driver IC.

In letteratura sono stati fatti tentativi che hanno cercato di alleviare parzialmente il problema di semplificazione della stima. Ad esempio, nell'articolo [36] si utilizza sempre una struttura a due pezzi ma si propone la stima dei parametri delle varie componenti osservando porzioni delle risposte in condizioni di funzionamento come mostrato nella Figura 4.1. In altri casi, come in [5] [13], si utilizzano modelli generali basati su una rete neurale per una stima simile dall'osservazione di risposte transitorie. Tuttavia, è importante notare che persistono alcune criticità tra cui la robustezza del modello stimato, la complessità (numero di parametri, dimensioni), la presenza di dinamiche spurie, comportamenti non fisici e l'implementazione SPICE.

Questo studio affronta ancora delle sfide significative. Si posiziona in un contesto che favorisce l'adozione dei modelli di regressione basata su kernel, come dimostrato dalla ricerca nel campo. Questi modelli sono preferibili per diversi motivi come la loro struttura lineare che semplifica il processo di addestramento anche quando si dispone di un set di dati di addestramento relativamente limitato.

Le sezioni successive di questo capitolo espandono ulteriormente questo argomento, dettagliando i setup proposti per la raccolta delle risposte transitorie necessarie per la stima dei modelli. Questa discussione include un'analisi dei carichi utilizzati durante le fasi di training e di test, spiegando come le risposte vengono utilizzate per creare un dataset da utilizzare nell'algoritmo di stima dei modelli KRR. Inoltre, vengono esaminati approfonditamente i metodi per l'utilizzo del modello stimato nelle simulazioni.

4.1 Modellazione di tipo NARX

Consideriamo il problema di descrivere in modo esplicito la dinamica di un elemento di circuito multiporta non lineare, con un vettore di ingresso $\boldsymbol{u}(t)$ e una singola uscita $y(t)$, attraverso un modello nella forma:

$$\hat{y}_k = f(\boldsymbol{u}_k, ..., \boldsymbol{u}_{k-r}, \hat{y}_{k-1}, ..., \hat{y}_{k-r}) \tag{4.2}$$

dove f è una funzione generica non lineare. Il vettore $\hat{y}_k$ rappresenta l'output stimato al tempo discreto k e $\mathbf{u}_k = [u_k^{(1)}, \ldots, u_k^{(n)}]^T$ rappresenta il vettore di input corrispondente, che contiene tutte le n tensioni e/o correnti delle porte del dispositivo al tempo discreto k a diversi istanti di tempo (ad esempio, $\hat{y}_k = \hat{y}(kT)$, dove T è il periodo di campionamento assunto). Il parametro r, invece, indica l'ordine del modello.

Questo modello è comunemente noto nella letteratura come modello **NOE** (Nonlinear Output Error) e si basa su un'equazione ricorsiva nell'output stimato. In pratica, la previsione del modello in un determinato istante dipende dalle previsioni negli r passaggi temporali precedenti. Tecniche di apprendimento automatico possono essere utilizzati per apprendere un modello NOE, come quello in 4.2, a partire da un insieme di input e output $\mathcal{D}_{NOE} = \{(\boldsymbol{x}_l, y_l)\}_{l=1+r}^{L}$, dove il vettore $\boldsymbol{x}_l = [\boldsymbol{u}_l, ..., \boldsymbol{u}_{l-r}]^T$ rappreseta i valori attuali e passati dell'input tempo discreto $\boldsymbol{u}_l$, e y_l è il segnale di output discreto corrispondente.

Nonostante le regressioni kernel siano strumenti di modellazione potenti, possono incontrare difficoltà nell'affrontare la struttura ricorsiva del modello NOE. L'impiego di questo tipo di regressioni, sebbene teoricamente fattibile, si scontra con notevoli sfide computazionali e di ottimizzazione, incluse la risoluzione di problemi non convessi [1]. Infatti, di soliti la mappa ricorsiva proposta in 4.2 viene appresa attraverso tecniche di regressione ricorrente, come le reti neurali ricorrenti (RNN).

Modelli di tipo autoregressivo non lineare con input esogeno (Nonlinear AutoRegressive eXogenous, **NARX**) sono solitamente usati per facilitare l'uso di regressioni con kernel in problemi di identificazione dei sistemi dinamici. Il modello NARX è descritto dalla seguente formulazione:

$$\hat{y}_k = f(\boldsymbol{u}_k, ..., \boldsymbol{u}_{k-r}, y_{k-1}, ..., y_{k-r}), \tag{4.3}$$

dove $(y_{k-1}, ..., y_{k-r})$ e $(\boldsymbol{u}_k, ..., \boldsymbol{u}_{k-r})$ rappresentano i **veri valori** dell'output e degli input in istanti di tempo diversi, e $\hat{y}_k$ è l'output stimato al tempo k.

[1] I problemi non convessi si verificano in situazioni di ottimizzazione caratterizzate da molteplici minimi locali, il che impedisce l'identificazione di una singola soluzione globale.

A differenza del modello NOE ricorsivo in 4.2, il modello NARX non presenta ricorsione nella variabile y_k, poiché utilizza come input i valori veri dell'output $(y_{k-1}, ..., y_{k-r})$ disponibili nell'insieme di addestramento.

Di conseguenza, il modello NARX in 4.3 è statico e può essere adeguatamente appreso attraverso regressioni kernel convenzionali utilizzando l'insieme di addestramento $\mathcal{D}_{NARX} = \{(\tilde{\boldsymbol{x}}_l, y_l)\}_{l=1+r}^{L_{NARX}}$, dove $\tilde{\boldsymbol{x}}_l = [\boldsymbol{x}_l, y_{l-1}, ..., y_{l-r}]^T$ contiene i valori attuali e passati dell'input di tempo discreto $\boldsymbol{x}_l$, mentre y_l rappresenta il vero output corripondente.

In questo contesto, tecniche di regressione con kernel come la KRR possono essere usate per imparare la mappa dinamica $f(\cdot)$ dell modello NARX. Il risultate modello può esere valutato in modo ricorrente ed ha la seguente struttura:

$$\hat{y}_k = \sum_{l=1+r}^{L} \alpha_l \boldsymbol{K}(\tilde{\boldsymbol{x}}_l, [\boldsymbol{x}_k, \hat{y}_{k-1}, ..., \hat{y}_{k-r}]) \tag{4.4}$$

dove $\{\alpha_l\}_{l=1+r}^{L}$ sono i coefficienti da stimare durante la fase di apprendimento e $K(\cdot, \cdot)$ è una funzione kernel scalare convenzionale.

I coefficienti $\boldsymbol{\alpha} = [\alpha_{1+r}, ..., \alpha_L]^T$ possono essere stimati in modo appropriato dall'insieme di addestramento $\mathcal{D}_{NARX}$ attraverso la soluzione del seguente sistema lineare:

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \tag{4.5}$$

dove $\boldsymbol{y} = [y_{1+r}, ..., y_L]$ sono i veri valori e $\boldsymbol{K}$ è il gramiano associato al kerel tale che $[K]_{ij} = k(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$ definita valutando la funzione kernel su ogni coppia di configurazioni nell'insieme di input di addestramento, λ è il paramatro di regolarizzazione (come descritto nel paragrafo 3.2.4) ed $\boldsymbol{I}$ è la matrice identità.

Il modello risultante può essere valutato come un modello ricorrente, il che significa che, nonostante sia stato addestrato come un modello statico, durante l'uso può essere interpretato e utilizzato come un modello che tiene conto delle informazioni passate. Questo è possibile grazie al fatto che il modello NARX include input e output passati nella sua struttura, consentendo al modello di "ricordare" le informazioni precedenti mentre fa previsioni per il futuro.

In questo studio, viene utilizzato un kernel a base radiale (RBF) gaussiano e i suoi iperparametri insieme a λ vengono stimati tramite una cross-validation a 3 fold. Il kernel RBF è preferito ad altre funzioni kernel all'avanguardia come il kernel polinomiale, poiché ha dimostrato prestazioni superiori in diversi problemi di regressione.

4.2 Carichi per l'addestramento e il Test

In questa sezione la tecnica di identificazione presentata in precedenza viene applicata ad generico buffer di uscita, costituito da un tipico driver CMOS, per il quale si utilizza una sequenza di bit di ingresso (bitstream) predefinita che asume i valori logici "010100". Tale sequenza logica è forzata dalla tensione di ingresso del buffer (si faccia riferimento alla tensione $v_{in}(t)$ nello schema di Fig. 4.1) che ha transizioni di livello logico trapezoidale.

Le variabili coinvolte sono rinominate come da schema di Fig. 4.2, con la tensione all'ingresso $v_1(t)$, la tensione alla porta di uscita $v_2(t)$ e la tensione alla porta relativa al nodo dell'alimentazione $v_3(t)$ scelte come ingressi. Le uscite sono le correnti $i_2(t)$ e $i_3(t)$. Facendo riferimento all'equazione (4.1), $\boldsymbol{u}(t) = [v_1(t), v_2(t), v_3(t)]^T$ e $\boldsymbol{y}(t) = [i_2(t), i_3(t)]^T$.

Le risposte transitorie di tensione e corrente sono campionate con campionamento uniforme mendiante un passo tenmporale T, producendo sequenze quali ad esempio $\{i_{1,1}, i_{1,2}, i_{1,3}, \dots i_{1,L}\}$ con $i_{1,k} = i_1(t = kT)$, con numero totale di campioni per ogni risposta pari ad L. Queste sequenze di dati potranno essere inseriti in vettori per confezionare i dati di ingresso e di uscita per la stima di modelli basati su regressione.

Nel lavoro si osserveranno le risposte del driver su un certo numero di carichi in modo da poter catturare le tipiche dinamiche di risposta del dispositivo in diverse situazioni operative, potenziando così la capacità del modello di machine learning di comprendere e prevedere il comportamento del circuito in contesti realistici.

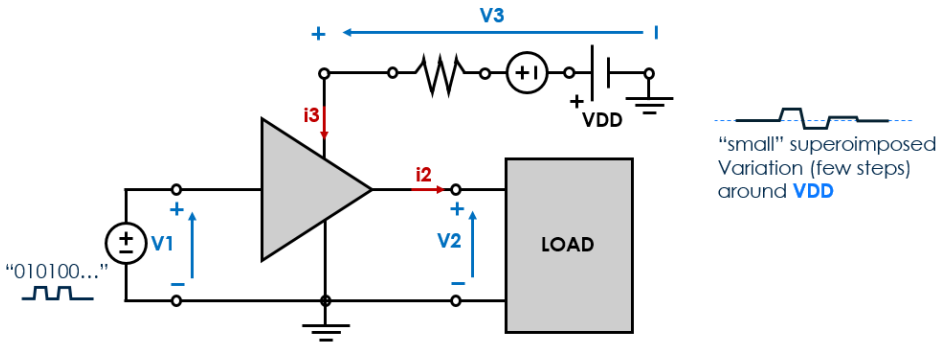

Figura 4.2: Set-up utilizzato.

Facendo riferimento al set-up di Fig. 4.2, i carichi considerati sono:

Test 1: in questo caso, il carico è scelto per rappresentare una idealizzazione di un canale di comunicazione digitale. Questo si ottiene considerando una linea di trasmissione ideale caratterizzata da un'impedenza caratteristica di 75 Ω e un ritardo temporale di 1.5 nanosecondi.

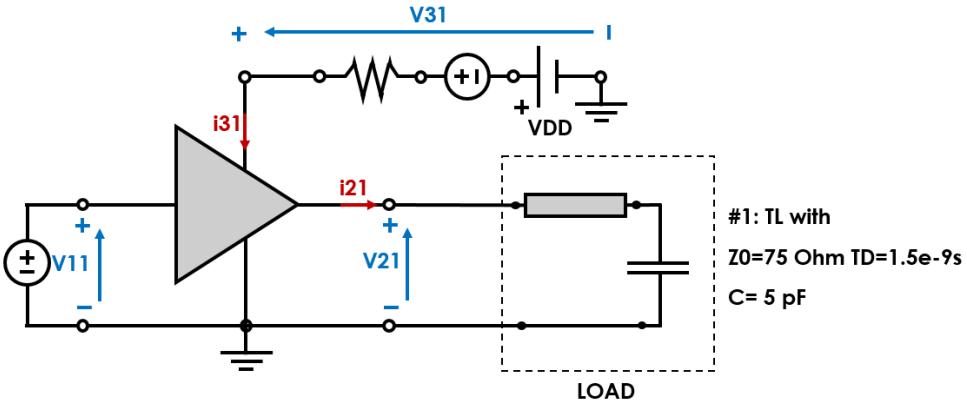

Figura 4.3: Test 1 - Linea di trasmissione #1

In figura 4.4 è riportata la netlist di tipo HSPICE di questo primo test case.

Si inizia definendo i parametri di simulazione, come PVDD, PVOUT, PVINH e PVINL. Per esempio, PVDD rappresenta la tensione di alimentazione nominale del circuito, mentre PVINH e PVINL indicano rispettivamente i livelli logici alto e basso del segnale.

La netlist include una libreria, "./driver.def", che conterrà la definizione del driver stesso. Questa libreria è cruciale poiché fornisce informazioni dettagliate sul comportamento e sulle caratteristiche del driver utilizzato nel circuito.

L'analisi del circuito è eseguita tramite il compando SPICE di analisi ".tran" che è annegato all'interno del sottocircuito in "InputBSandTRAN.lib", che fornisce anche l'onda trapezioidale per la generazione della stringa di bit scelta.

Il driver è costituito da componenti come sorgenti di tensione e condensatori, ai quali sono stati collegati gli elementi che simulano la linea di trasmissione.

Alla fine, il comando ".probe" imposta le sonde di simulazione per monitorare le tensioni e le correnti ai nodi di interesse.

```
EXAMPLE SIM (simplest possible SE driver)

* parameters

  .PARAM PVDD  = 1.8
  .PARAM PVOUT = 0
  .PARAM PVINH = 1.8
  .PARAM PVINL = 0

  .include "./driver.def"

  .Option CSDF = 1
  .Option probe

* enable
V55 55 0 0

* power supply
V33 33 0 'PVDD'


.inc InputBSandTRAN.lib

* driver switching on a TL load
  XS1 1 777 0 MYVIN

  XD1 1 2 3 0 55 DRIVER
  V21 2 4 0
  C21 2 0 5e-12
  T41 4 0 6 0 Z0=75 TD=2e-9
  C41 4 0 5e-12

* power supply

  V31 3 33 PWL (0 0 50n 0 55n 0.2 65n 0.2 70n -0.2 90n -0.2 95n 0.1 100n
+ 0.1)

.probe V(1) V(2) V(3) I(V21) I(V31)

.end
```

Figura 4.4: Netlist associata al Test 1

Test 2: in questo caso, la presenza di un resistore da 50 Ω in parallelo a un condensatore da 10 pF simula l'aggiunta di un carico capacitivo al circuito. Questo tipo di carico è comune nelle applicazioni reali quando la interconnessione é corta.

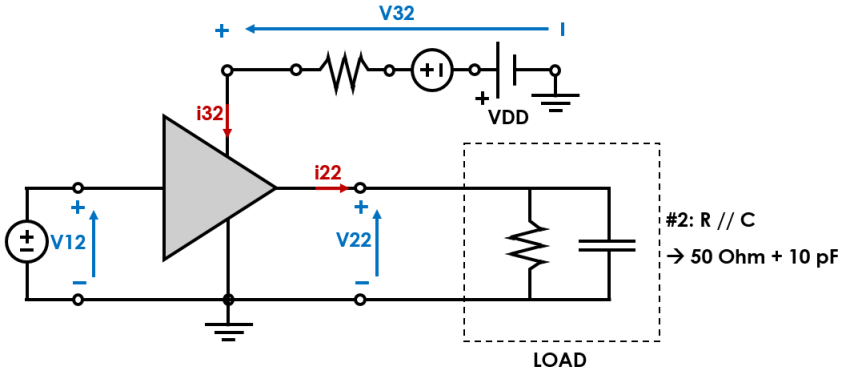

Figura 4.5: Test 2 - Carico RC

Test 3: il carico è costituito dalla connessione serie di un resistora da 50 Ω con un generatore ideale di tensione che forza la tensione nominale di alimentazione VDD.

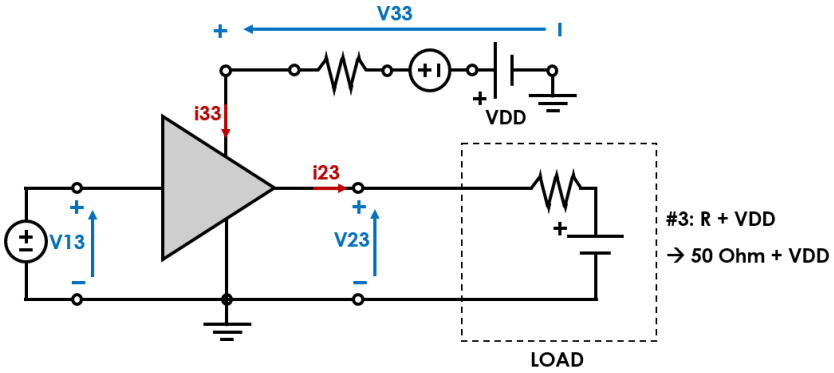

Figura 4.6: Test 3 - Carico resistivo in serie alla alimentazione

Test 4: similmente al caso del Test 3, si ha nuovamente un carico resistivo ma con valore differente di resistenza (150 Ω) e senza l'aggancio alla tensione di alimentazione.

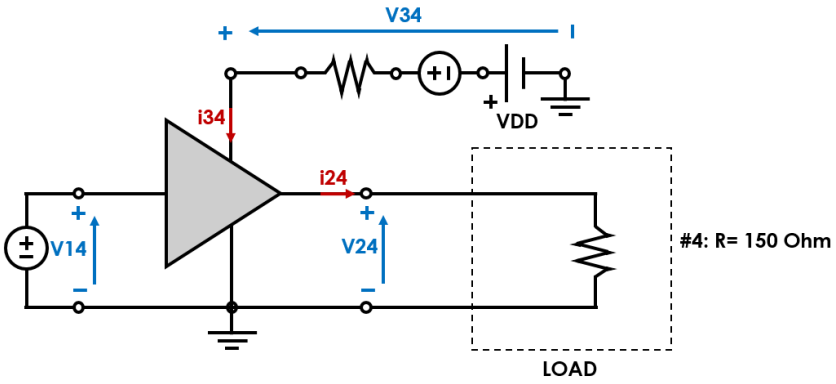

Figura 4.7: Test 4 - Carico resistivo

Test 5: infine, questo test è simile al primo ma presenta una linea di trasmissione differente, con un'impedenza caratteristica di 50 Ω e un ritardo temporale di 2 nanosecondi. Ciò consente di esplorare gli effetti delle variazioni nei parametri della linea di trasmissione sul comportamento del circuito e permettendo di definire un carico per la validazione o test del modello.

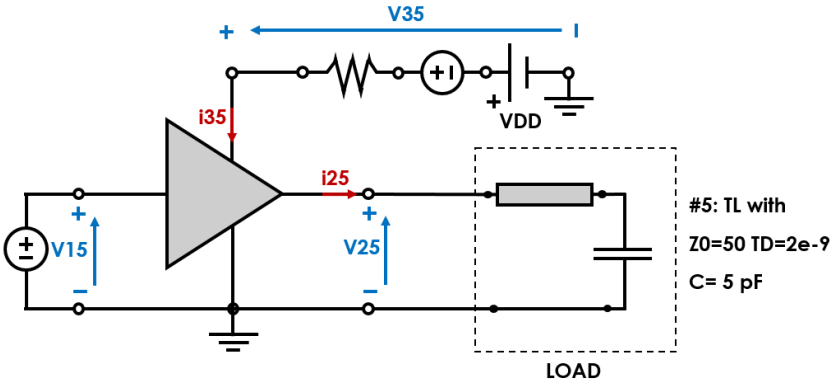

Figura 4.8: Test 5 - Linea di trasmissione #2

Ogni set di carico è associato a un file differente, simulato con un comando all'interno dell'ambiente MATLAB che richiama il simulatore esterno SPICE. Per ciascun file di simulazione, i dati vengono estratti e organizzati in una **struttura dati** appositamente progettata. Tale struttura mira a conservare i dati relativi a

ogni variabile di interesse, tensioni e correnti, al fine di renderli facilmente accessibili per le successive analisi.

Come brevemente accennato prima, successivamente all'estrazione, i dati sono sottoposti a un processo di **resampling**, con un tempo di campionamento (T) predefinito, per assicurare un campionamento uniforme nel tempo. Questo aspetto è cruciale per garantire una rappresentazione accurata dei dati e per agevolare le successive analisi e comparazioni tra i diversi set di dati.

Tutte le variabili di interesse disposte in colonne vengono **aggregati in un unico pacchetto**, in modo sequenziale per ogni istante temporale. Questa formattazione tabulare dei dati semplifica ulteriormente l'analisi e la manipolazione dei dati stessi. In forma matriciale, L campioni delle singole risposte associate ai set usati per l'addestramento del modello saranno memorizzati nella matrice $\boldsymbol{D}$:

$$\boldsymbol{D} = \left[\begin{array}{ccc|cc} v_{11,1} & v_{21,1} & v_{31,1} & i_{21,1} & i_{31,1} \\ v_{11,2} & v_{21,2} & v_{31,2} & i_{21,2} & i_{31,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{11,L} & v_{21,L} & v_{31,L} & i_{21,L} & i_{31,L} \\ \hline v_{12,1} & v_{22,1} & v_{32,1} & i_{22,1} & i_{32,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline v_{13,1} & v_{23,1} & v_{33,1} & i_{23,1} & i_{33,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline v_{14,1} & v_{24,1} & v_{34,1} & i_{24,1} & i_{34,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right] \tag{4.6}$$

È da notare che tale aggregazione è stata eseguita esclusivamente per le variabili raccolte nei primi quattro set di carico, poiché rappresentano i valori utilizzati nella fase di addestramento, tutti questi valori vanno a formare il **Data Set**. Le variabili associate al quinto set sono state raccolte in una matrice separata $\boldsymbol{T}$, impiegata successivamente nella fase di test del modello (**Test Set**):

$$\boldsymbol{T} = \left[\begin{array}{ccc|cc} v_{15,1} & v_{25,1} & v_{35,1} & i_{25,1} & i_{35,1} \\ v_{15,2} & v_{25,2} & v_{35,2} & i_{25,2} & i_{35,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{15,L} & v_{25,L} & v_{35,L} & i_{25,L} & i_{35,L} \end{array}\right] \tag{4.7}$$

Prima di procedere con l'addestramento del modello di regressione, i dati vengono opportunamente preparati.

E' stata prima utilizzata una funzione la cui parte principale coinvolge la manipolazione dei dati nelle matrici $\boldsymbol{D}$ e $\boldsymbol{T}$. Utilizzando l'ordine r specificato, vengono ritardati i valori delle variabili di input e output per creare una struttura che tenga conto delle dipendenze temporali o sequenziali nei dati, ad esempio:

$$\boldsymbol{x}_l = \begin{bmatrix} v_{1,l+r} \\ v_{1,l+(r-1)} \\ v_{1,l+(r-2)} \\ \vdots \\ v_{1,l} \\ \hline v_{2,l+r} \\ v_{2,l+(r-1)} \\ \vdots \\ v_{2,l} \\ \hline v_{3,l+r} \\ v_{3,l+(r-1)} \\ \vdots \\ v_{3,l} \end{bmatrix} \tag{4.8}$$

e

$$\boldsymbol{y}_l = \begin{bmatrix} i_{2,l+r} \\ i_{2,l+(r-1)} \\ \vdots \\ i_{2,(l-1)} \\ \hline i_{3,l+r} \\ i_{3,l+(r-1)} \\ \vdots \\ i_{3,(l-1)} \end{bmatrix} \tag{4.9}$$

Capitolo 5

Analisi dei risultati ottenuti

La parte forse più importante di qualsiasi modello di Machine Learning corrisponde alla misura delle sue prestazioni, in cui si valuta la capacità predittiva e la robustezza che ha acquisito.

Nel capitolo 3 sono state illustrate varie tecniche di addestramento, concentrandosi specificamente sulle metodologie di regressione basate sul Kernel. Nel capitolo 4, invece, sono state approfondite le strategie per l'applicazione di tali tecniche, insieme alla valutazione dei differenti set di dati che si intendono utilizzare.

Questo capitolo si addentra nella descrizione del componente reale utilizzato, con il quale vengono catturate e tracciate le variazioni delle variabili elettriche, fondamentali per la costruzione delle matrici di Data Set e di Test Set.

Successivamente, con il modello allenato, si procede alla valutazione dei risultati attraverso l'applicazione di due approcci distinti: uno di natura statica e l'altro ricorrente. Questo approccio permette di valutare la accuratezza e le caratteristiche di risposta del modello sia rispetto alla caratteristica dinamica sia alla componente statica, fornendo così una visione completa delle sue capacità predittive.

5.1 Dispositivo reale scelto

Per condurre lo studio, è stato impiegato un componente reale, disponibile in commercio, noto come "SN74ALVCH16973" che fa parte della famiglia di circuiti integrati logici CMOS del produttore Texas Instruments. La scheda tecnica (datasheet) del dispotivo e il suo modello di tipo Transistor-level possono essere scaricati dal sito del produttore, all'indirizzo https://www.ti.com/product/SN74ALVCH16973.

Questo componente offre tempi di propagazione bassi e tempi di commutazione ridotti, consentendo il trasferimento rapido dei dati attraverso canali di comunicazione. Il SN74ALVCH16973 è caratterizzato da 4 buffer indipendenti e 8 transceiver, permettendo la gestione di segnali digitali. Questa funzionalità lo rende particolarmente adatto per applicazioni che richiedono il trasferimento di dati su più linee; applicazioni che spaziano da quelle automotive a quelle biomedicali. Inoltre, è progettato per garantire elevate prestazioni con un basso consumo energetico. La sua versatilità è dimostrata dalla sua capacità di adattarsi a un'ampia gamma di tensioni di alimentazione, che variano da 1.65 V a 3.6 V. Nel contesto di questa ricerca, è stata utilizzata una tensione di alimentazione nominale del componente pari a 1.8 V.

Il dispositivo é stato simulato nel simulatore HSPICE, per ottenere le risposte di riferimento impiegate per la stima del modello (training) e per la sua valitazione (test).

5.2 Segnali di Training e di Test

Il componente precedentemente descritto è stato quindi impiegato per generare le sue risposte a ciascun set di carico (come specificato in 4.2). I risultati di queste simulazioni hanno fornito i valori effettivi delle tensioni e delle correnti, che sono stati impiegati come dati di input e output per il modello. E' stata utilizzata anche nella seguente applicazione, la sequenza di bit di ingresso con i seguenti valori logici "010100".

La Figura 5.1 mostra le risposte di tensone e corrente associate ai primi quattro test $k = 1, \ldots, 4$. Ogni test è associato a un pannello in figura, composto da due subplot.

- Nel primo subplot (in alto), le risposte delle tensioni (in V) nel tempo (in ns) sono rappresentate come segue.
  - $v_{1k}(t)$: tensione di alimentazione, indicata da una linea nera tratteggiata.
  - $v_{2k}(t)$: tensione della porta di uscita, indicata da una linea blu continua.
  - $v_{3k}(t)$: tensione di alimentazione del dispositivo, indicata da una linea magenta continua.

- Nel secondo subplot (in basso), le risposte delle correnti (in mA) nel tempo (in ns) sono rappresentate come segue.
  - $i_{2k}(t)$: corrente che esce dal terminale associato alla porta di uscita, rappresentata da una linea blu continua.
  - $i_{3k}(t)$: corrente che entra nel dispositivo dal terminale di alimentazione, indicata da una linea magenta continua.

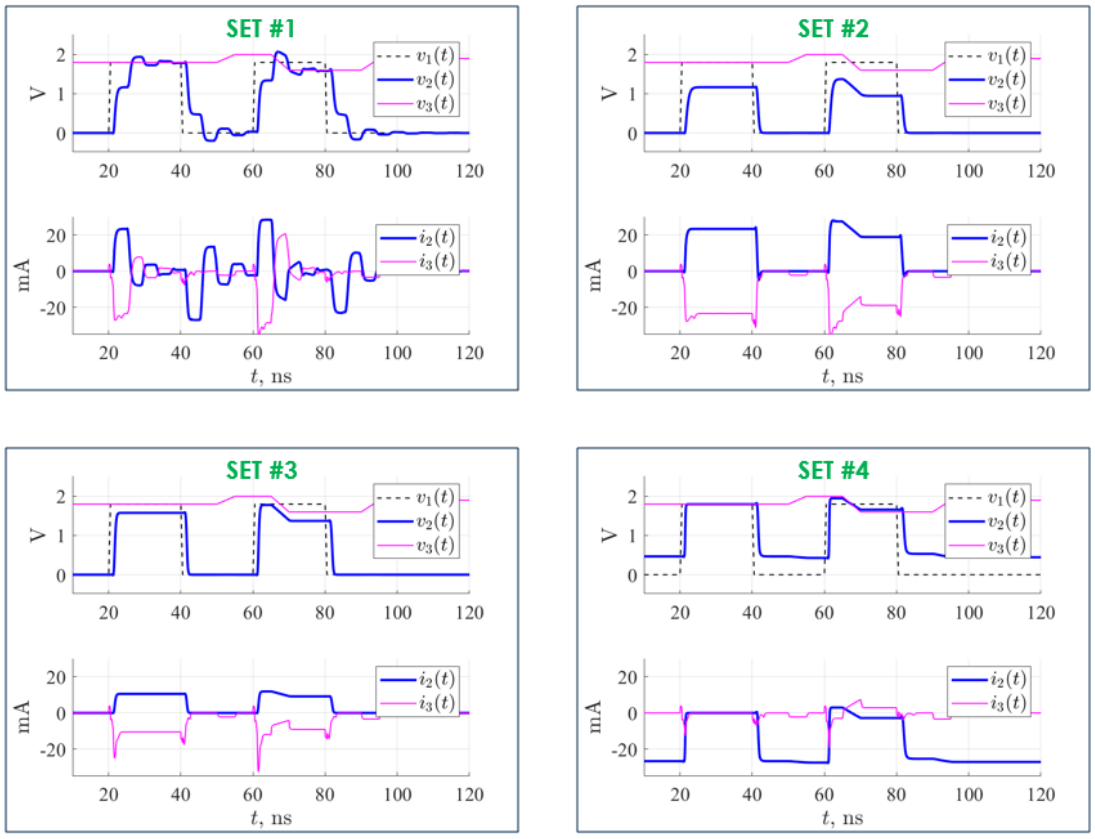

Figura 5.1: Risposte transitorie delle tensioni e delle correnti coinvolte osservate quando il dispositivo di test selezionato opera nelle 4 condizioni di carico usate per la stima. Test #1: linea di trasmissione con Z0=75 e TD=1.5e-9; #2: connessione shunt 50 Ohm // 10pF; #3: 50 Ohm in serie con la batteria di alimentazione VDD; #4: 150 Ohm.

La Figura 5.2 mostra invece le risposta del dispositivo relative al quinto test, quello utilizzato durante la fase di test. Anche in questo caso, è stata utilizzata la stessa rappresentazione grafica, con le medesime distinzioni di colore e forma delle linee per una chiara identificazione delle variabili coinvolte.

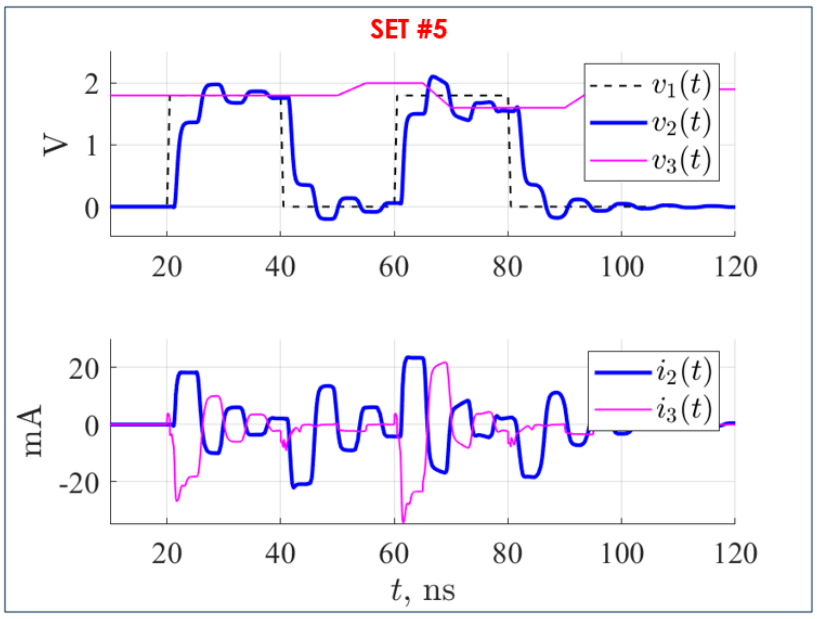

Figura 5.2: Risposte transitorie delle tensioni e correnti coinvolte che sono associate al test #5: linea di trasmissione con Z0=50 e TD=2e-9.

5.3 Modello statico vs ricorrente

Dopo aver raccolto le risposte di tensione e corrente del dispositivo, si è seguito il procedimento descritto nella sezione 4.2, raccogliendo tali risposte nelle matrice del Data Set. Successivamente, utilizzando tale matrice e le tecniche KRR, è stato eseguito il training per la stima del modello. É importante notare che il tempo necessario per la stima del modello é di soli 104 secondi, tempo trascurabile se si considera l'intero processo che prevede una sola volta la stima dei parametri del modello. La fase successiva dell'analisi ha comportato la simulazione del modello addestrato al fine di valutarne l'accuratezza nel riprodurre il comportamento del componente.

In dettaglio, il modello è stato testato utilizzando due metodologie correlate, che differiscono nel loro approccio alla gestione dei dati storici. Questi metodi sono stati implementati in MATLAB attraverso la creazione di due funzioni distinte, identificate come 'static' e 'recurrent'. Entrambe le funzioni utilizzano parametri specifici del modello e i valori contenuti nel Test Set. La seconda funzione introduce anche parametri aggiuntivi come il ritardo (lag) e l'ordine del modello (r), che influenzano la lunghezza della finestra di dati storici utilizzata per fare previsioni future.

La prima funzione, 'static', fa riferimento all'equazione 4.4. Produce una stima diretta basata esclusivamente sui dati forniti, senza tener conto dell'evoluzione nel tempo dei dati istante per istante, infatti, viene applicata una sola volta.

D'altra parte la seconda funzione, 'recurrent', prende in considerazione l'evoluzione nel tempo dei dati storici istante per istante per fare previsioni future. Nel caso di questa funzione, viene mantenuta una finestra di dati storici (veri valori dell'output) di lunghezza lag e basandosi su questi dati si procede in modo iterativo a calcolare l'otuput all'istante successivo utilizzando la funzione 'static'.

$$\hat{y}_k = \begin{cases} y_k & \text{se } 1 < k \leq \text{lag} \\ \sum_{l=1+r}^{L} \alpha_l \boldsymbol{K}(\tilde{\mathbf{x}}_l, [\mathbf{x}_k, \hat{y}_{k-1}, \ldots, \hat{y}_{k-r}]) & \text{se } k > \text{lag} \end{cases} \tag{5.1}$$

Questo secondo approccio é necessario quando si dovrá implementare il modello in un ambiente di simulazione quale SPICE. Per il caso in esame, il ritardo iniziale lag é stato scelto pari a $(r + 1)$. L'ordine dinamico scelto é invece $r = 6$.

5.4 Validazione

Nella fase di test, entrambi gli approcci descritti, sia 'static' che 'recurrent', sono stati impiegati per ottenere un confronto diretto sui risultati. I dati sono stati analizzati separatamente per ciascuna variabile di output. Nelle rappresentazioni grafiche riportate di seguito sono state utilizzate diverse linee per evidenziare le differenze:

- Linea nera continua: rappresenta i valori reali di riferimento;
- Linea rossa tratteggiata: indica i risultati ottenuti tramite l'approccio 'static';
- Linea blu tratteggiata: indica i risultati ottenuti tramite l'approccio 'recurrent'.

Questa notazione è stata mantenuta costante per entrambe le rappresentazioni.

Nella figura 5.3, sono riportati gli andamenti della corrente $i_2(t)$. I risultati dimostrano un'eccellente corrispondenza della corrente di uscita nei punti di transizione e negli overshoots [1] per il modello ricorrente. Per il modello statico, non si è verificata alcuna perdita di precisione.

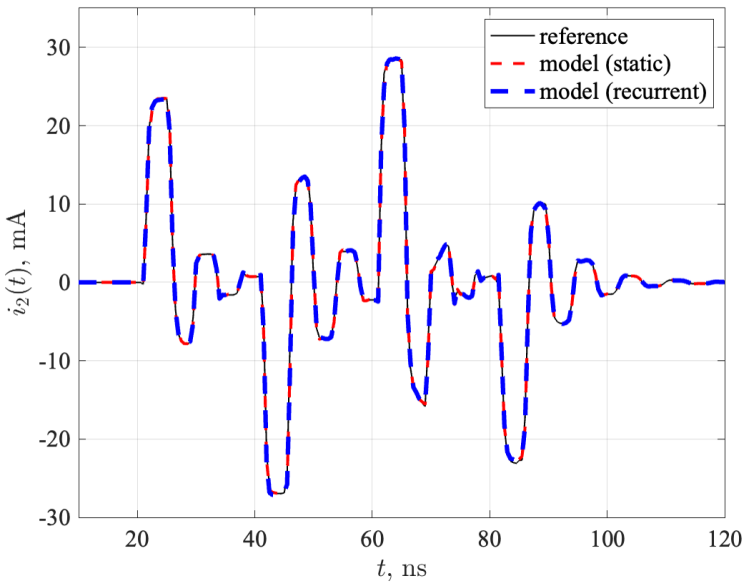

Figura 5.3: Validazione del modello sulla corrente di output di riferimento, $i_2(t)$, ottenuta sia dalla versione statica del modello che dalla versione ricorrente.

[1] Gli overshoots sono variazioni temporanee che portano una grandezza oltre un limite o un obiettivo desiderato prima di tornare ai valori stabili.

In figura 5.4 sono riportati i risultati relativi alla risposta della corrente di alimentazione $i_3(t)$, confermando anche per questa risposta l'elevata accuratezza che si ottiene sia con l'approccio statico che riccorente.

La simulazione del modello di tipo 'static' e quello di tipo 'recurrent' nell'ambiente Matlab é stata in entrambi i casi meno di $1s$.

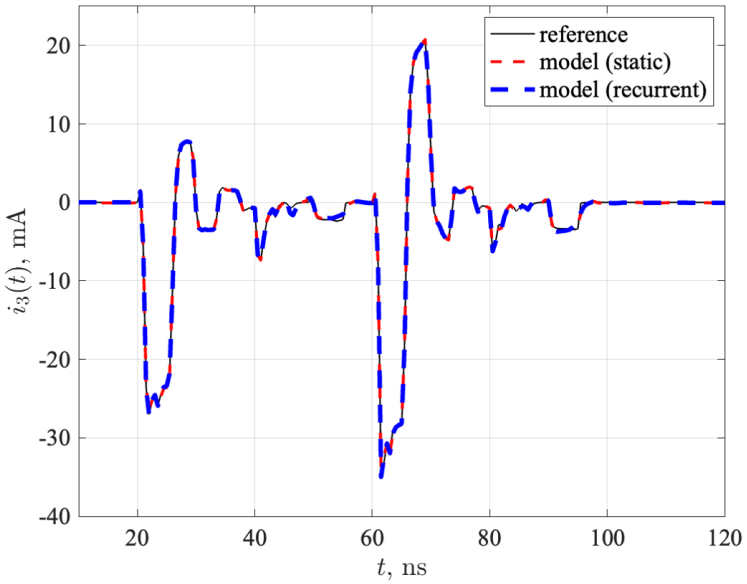

Figura 5.4: Validazione del modello sulla corrente di alimentazione, $i_3(t)$, ottenuta sia dalla versione statica del modello che dalla versione ricorrente.

Le curve riportate in figura rappresentano la caratteristica dinamica del componente, evidenziando l'andamento dei parametri in funzione del tempo. Questi andamenti sono cruciali poiché forniscono informazioni sulle transizioni tra gli stati logici e sulla velocità di risposta del sistema.

D'altra parte, la risposta statica del buffer si concentra sulla stabilità del segnale quando si trova in uno stato logico definito, sia alto che basso. Per validare il modello e condurre un'analisi più approfondita, è stato dunque scelto di simulare il comportamento del sistema in condizioni statiche.

Anche in questa situazione i dati reali utilizzati per la verifica sono stati ricavati mediante simulazione SPICE. Il nucleo della simulazione si è concentrato sul comportamento del driver nei due stati fondamentali: alto (H) e basso (L). Nel dettaglio, si sono impostate le tensioni di ingresso corrispondenti agli stati H e L, utilizzando istruzioni specifiche per modellare il comportamento del driver in ognuno di essi. In seguito, è stata eseguita una simulazione DC per vari valori di

tensione di ingresso, e tramite delle sonde di prova si sono monitorate e registrate le tensioni e le correnti nei punti specifici del circuito.

Le sequenze temporali di tensione e corrente $v_2(t)$ e $i_2(t)$ sono state rappresentate come traiettorie nel piano (v, i) nella figura 5.5. Viene mostrato come i diversi set utilizzati nella fase di addestramento abbiano consentito al modello di esplorare densamente le varie aree di lavoro, migliorando così le prestazioni nelle previsioni.

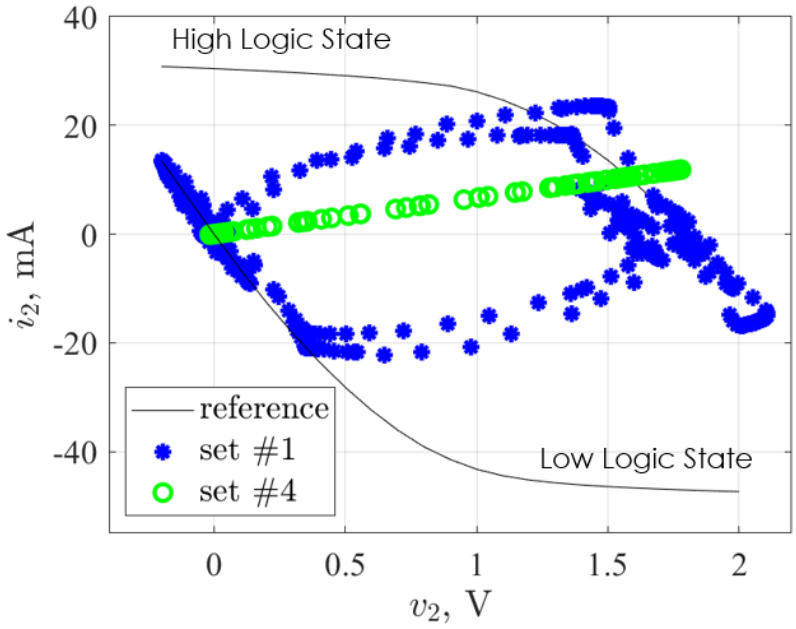

Figura 5.5: Caratteristica statica del componente con le zone esplorate dal: test #1: linea di trasmissione con Z0=75 e TD=1.5e-9 e test #4: 150 Ohm.

Anche in questo caso, il modello è stato validato utilizzando entrambi gli approcci 'static' e 'recurrent', identificati in figura con la stessa notazione (colore e spessore).

Nella figura 5.6, é evidente che il modello presenta un'eccellente corrispondenza nelle regioni statiche di interesse, esplorate dai segnali di training (indicate in giallo) per i diversi stati logici, mentre si verifica una discrepanza nelle regioni non esplorate (indicate in grigio). In queste zone grigie non vi è però alcuna criticitá in quanto si tratta di zone che non verranno eplorate durante il funzonamento del mdoello in una simulazione circuitale in condizioni di applciazione reale.

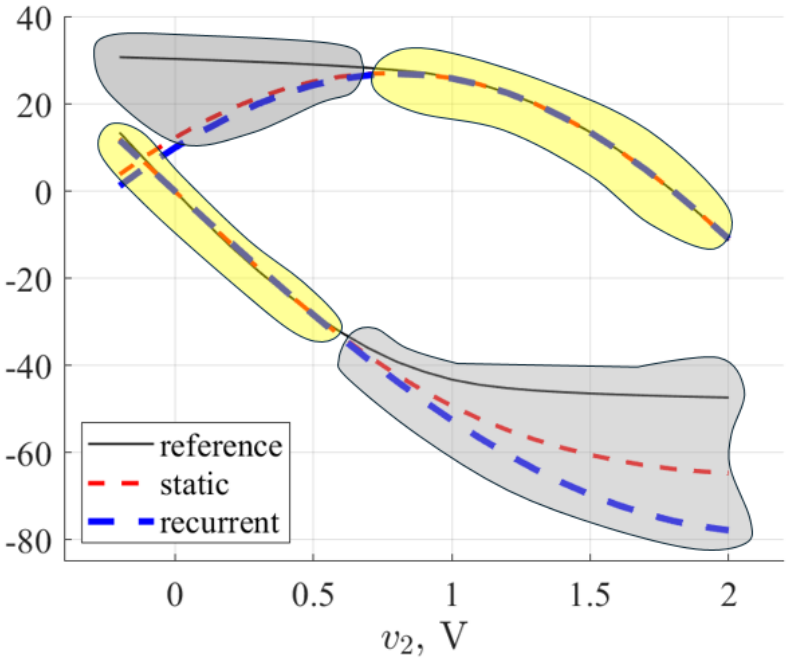

Figura 5.6: Validazione del modello sulla caratteristica statica.

Capitolo 6

Conclusioni e sviluppi futuri

Il lavoro di ricerca svolto in questa tesi si é concentrato sulla modellazione comportamentale dei buffer dei circuiti digitali integrati, con l'obiettivo di valutare le prestazioni dei collegamenti dati ad alta velocità e introduce un'innovativa tecnica di modellazione che offre diversi vantaggi significativi. Inoltre, l'approccio proposto rispetta tutti i requisiti dell'identificazione black-box: basandosi esclusivamente sull'osservazione delle tensioni e delle correnti alle porte esterne dei dispositivi, consente l'estrazione di modelli che replicano il loro funzionamento senza richiedere conoscenza della struttura interna. Inoltre, a differenza degli algoritmi standard attualmente utilizzati, il metodo descritto in questo studio offre una soluzione diretta per la modellazione del comportamento ingresso-uscita, offrendo un primo sforzo di generalizzazione.

Un aspetto cruciale del lavoro è stata la validazione del modello di predizione, focalizzandosi sulla sua capacità di gestire in modo completamente automatico e scalabile più uscite, includendo la corrente di alimentazione, e più ingressi, includendo anche la tensione di ingresso dei buffer. Questo ampliamento della capacità del modello mira a migliorare la sua versatilità e la sua accuratezza nelle previsioni, particolarmente rilevante in scenari realistici.

Nel corso della mia indagine, ho esaminato una vasta gamma di approcci, dalle tecniche standard a quelle più avanzate, alla ricerca delle soluzioni più idonee. É emerso che l'impiego del machine learning offre diverse risposte ai problemi esistenti, consentendo una valutazione tempestiva delle prestazioni del sistema fin dalle fasi iniziali di sviluppo.

Un ostacolo significativo della progettazione elettronica moderna è rappresentato dalle frequenti modifiche di progetto, spesso causate dalla crescente complessità hardware. Queste modifiche, in gran parte, derivano da una modellazione insufficiente, che rende le simulazioni lente e dispendiose. Con il continuo aumento della complessità e delle prestazioni, qualsiasi approssimazione o assunzione può portare a errori significativi. In queste circostanze, l'eliminazione dell'intervento

umano a favore del lavoro automatizzato può offrire vantaggi considerevoli. Proprio per superare queste sfide, sono state proposte soluzioni che puntano allo sviluppo di un paradigma di progettazione basato su modelli che imparano velocemente, sostituendo il tradizionale modello "lento" nella progettazione e nell'ottimizzazione. Il macromodeling, che impiega strumenti matematici e software per creare modelli surrogati (come i metodi KRR analizzati), é cruciale per gestire la crescente complessità dei sistemi e garantire prestazioni elevate nei dispositivi elettronici moderni, offrendo un metodo potente ed efficiente per affrontare le sfide della progettazione e dell'ottimizzazione.

I passi futuri saranno rivolti ad ottimizzare sia il periodo di campionamento che l'ordine dinamico del modello durante la fase di addestramento, e valutare le migliori implementazioni del modello direttamente nell'ambiente di lavoro SPICE. Inoltre, si potrá applicare lo strumento a tecnologie differenti quali le logiche differenziali ed un altro campo di applicazione per verificare se in quel caso i classificatori hanno prestazioni affini. Tutto ciò non solo punta a migliorare la comprensione dei buffer digitali, ma anche a gettare le basi per applicazioni future in diversi contesti.

Appendice A

Differenziazione di matrici

Questo insieme di equazioni introduce un vettore $\boldsymbol{x}$ di lunghezza m e una matrice $\boldsymbol{A}$ di dimensioni n×m, presupponendo che siano perpendicolari. Si esplorino le variazioni del vettore $\boldsymbol{y}$ rispetto a $\boldsymbol{x}$ in diverse situazioni.

$$\boldsymbol{x} = [m, 1] \tag{A.1}$$
$$\boldsymbol{A} = [n, m] \tag{A.2}$$

Le equazioni delineano varie situazioni di derivazione di $\boldsymbol{y}$ rispetto a $\boldsymbol{x}$ in contesti differenti. Ad esempio, quando $\boldsymbol{y}$ è il prodotto di $\boldsymbol{A}$ e $\boldsymbol{x}$, la variazione di $\boldsymbol{y}$ rispetto a $\boldsymbol{x}$ è data dalla matrice $\boldsymbol{A}$ stessa. In altre situazioni, come quando $\boldsymbol{y}$ è il prodotto di $\boldsymbol{x}$ e $\boldsymbol{A}$, la variazione di $\boldsymbol{y}$ rispetto a $\boldsymbol{x}$ coinvolge la trasposta di $\boldsymbol{A}$. Questa diversità nelle derivazioni riflette le varie interazioni tra $\boldsymbol{x}$, $\boldsymbol{y}$ e $\boldsymbol{A}$ in contesti specifici.

$$\boldsymbol{y} = \boldsymbol{A} \rightarrow \frac{\delta \boldsymbol{y}}{\delta \boldsymbol{x}} = 0 \tag{A.3}$$
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \rightarrow \frac{\delta \boldsymbol{y}}{\delta \boldsymbol{x}} = \boldsymbol{A} \tag{A.4}$$
$$\boldsymbol{y} = \boldsymbol{x}\boldsymbol{A} \rightarrow \frac{\delta \boldsymbol{y}}{\delta \boldsymbol{x}} = \boldsymbol{A}^T \tag{A.5}$$
$$\boldsymbol{y} = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \rightarrow \frac{\delta \boldsymbol{y}}{\delta \boldsymbol{x}} = 2\boldsymbol{x}^T\boldsymbol{A} \tag{A.6}$$

Bibliografia

[1] M. Swaminathan, C. Daehyun, S. Grivet-Talocia, K. Bharath, V. Laddha e X. Jianyong. «Designing and Modeling for Power Integrity». In: *IEEE Transactions on Electromagnetic Compatibility* 52 (2010), pp. 288–310 (cit. a p. 2).

[2] J. Fan, X. Ye, J. Kim, B. Archambeault e A. Orlandi. «Signal integrity Design for High-Speed Digital Circuits: Progress and Directions». In: *IEEE Transactions on Electromagnetic Compatibility* 52.2 (mag. 2010), pp. 392–400 (cit. alle pp. 2, 11).

[3] M. S. Sharawi e D. N. Aloi. «Circuit Modeling in High-Speed Designs». In: *IEEE Potentials* 24.1 (2005), pp. 17–20 (cit. alle pp. 2, 5, 6).

[4] *IBIS (I/O Buffer Information Specification) – Ver. 6.1*. Available online. `https://ibis.org/`. 2015 (cit. alle pp. 5, 6, 35).

[5] Yi Cao e Q. J. Zhang. «A New Training Approach for Robust Recurrent Neural-Network Modeling of Nonlinear Circuits». In: *IEEE Transactions on Microwave Theory and Techniques* 57.6 (2009), pp. 1539–1553 (cit. alle pp. 5, 9, 10, 36).

[6] A. K. Varma, M. Steer e P. D. Franzon. «Improving Behavioral IO Buffer Modeling Based on IBIS». In: *IEEE Transactions on Advanced Packaging* 31.4 (2008), pp. 711–721 (cit. alle pp. 5, 9).

[7] T. Zhu, M. B. Steer e P. D. Franzon. «Accurate and Scalable IO Buffer Macromodel Based on Surrogate Modeling». In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 1.8 (2011), pp. 1240–1249 (cit. a p. 9).

[8] M. B. Yelten, T. Zhu, S. Koziel, P. D. Franzon e M. B. Steer. «Demystifying Surrogate Modeling for Circuits and Systems». In: *IEEE Circuits and Systems Magazine* 12.1 (2012), pp. 45–63 (cit. a p. 9).

[9] W. Dghais, T. R. Cunha e J. C. Pedro. «Reduced-Order Parametric Behavioral Model for Digital Buffers/Drivers With Physical Support». In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 2.12 (2012), pp. 2071–2079 (cit. a p. 9).

[10] W. Dghais, H. M. Teixeira, T. R. Cunha e J. C. Pedro. «Novel Extraction of a Table-Based I–Q Behavioral Model for High-Speed Digital Buffers/Drivers». In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 3.3 (2013), pp. 500–507 (cit. a p. 9).

[11] W. Dghais, T. R. Cunha e J. C. Pedro. «A Novel Two-Port Behavioral Model for I/O Buffer Overclocking Simulation». In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 3.10 (2013), pp. 1754–1763 (cit. a p. 9).

[12] W. Dghais e J. Rodriguez. «New Multiport I/O Model for Power-Aware Signal Integrity Analysis». In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* (2016). (in press) (cit. a p. 9).

[13] Yi Cao, R. Ding e Q. J. Zhang. «State-space dynamic neural network technique for high-speed IC applications: modeling and stability analysis». In: *IEEE Transactions on Microwave Theory and Techniques* 54.6 (2006), pp. 2398–2409 (cit. alle pp. 10, 36).

[14] I. S. Stievano, I. A. Maio e F. G. Canavero. «M[pi]log, macromodeling via parametric identification of logic gates». In: *IEEE Transactions on Advanced Packaging* 27.1 (feb. 2004), pp. 15–23 (cit. alle pp. 10, 35).

[15] G. Signorini, C. Siviero, S. Grivet-Talocia e I. S. Stievano. «Power and Signal Integrity co-simulation via compressed macromodels of high-speed transceivers». In: *Proc. of the 2015 IEEE 18th Workshop on Signal and Power Integrity (SPI)*. Berlin, Germany, mag. 2015 (cit. alle pp. 10, 35).

[16] B. Mutnury, M. Swaminathan e J. P. Libous. «Macromodeling of nonlinear digital I/O drivers». In: *IEEE Transactions on Advanced Packaging* 29.1 (2006), pp. 102–113 (cit. a p. 10).

[17] C. Diouf, M. Telescu, I. S. Stievano, N. Tanguy e F. G. Canavero. «Simplified topology for IC buffer behavioral models». In: *IET Circuits, Devices & Systems* (2016). (in press) (cit. a p. 10).

[18] T. R. Cunha, H. M. Teixeira, J. C. Pedro, I. S. Stievano, L. Rigazio, F. G. Canavero, A. Girardi, R. Izzi e F. Vitale. «Validation by Measurements of an IC Modeling Approach for SiP Applications». In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 1.8 (2011), pp. 1214–1225 (cit. a p. 11).

[19] Tullio De Mauro. *Grande dizionario italiano dell'uso*. Torino: UTET, 2000 (cit. a p. 12).

[20] John McCarthy. *WHAT IS ARTIFICIAL INTELLIGENCE?* Stanford: Stanford University Press, 2004 (cit. a p. 12).

[21] European Commission. *Communication from the Commission to the European Parliament, the European Council, the Council, the European Economic and Social Committee and the Committee of the Regions on Artificial Intelligence for Europe*. COM 2018. Brussels, 237 final (cit. a p. 13).

[22] Stuart Russell e Peter Norvig. *Intelligenza Artificiale, un approccio moderno*. Vol. 1. Milano: Pearson Education Italia, 2005 (cit. a p. 13).

[23] SAS. *SAS Viya*. 2020. URL: https://www.sas.com/it_it/insights/analytics/what-is-artificial-intelligence.html#:~:text=Come%20funziona%20l'Intelligenza%20Artificiale,o%20dalle%20caratteristiche%20dei%20dati (cit. a p. 13).

[24] Terence Milles. «Machine Learning Vs. Artificial Intelligence: How Are They Different?» In: *Forbes* (2018). URL: https://www.forbes.com/sites/forbestechcouncil/2018/07/11/machine-learning-vs-artificial-intelligence-how-are-they-different/?sh=6d8d39173521 (cit. a p. 14).

[25] Arthur L. Samuel. «Some Studies in Machine Learning Using the Game of Checkers». In: *IBM Journal of Research and Development* (1959) (cit. a p. 14).

[26] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997 (cit. alle pp. 14, 16).

[27] Richard O. Duda, Peter E. Hart e David G. Stork. *Pattern Classification*. New York: John Wiley & Sons, Inc., 2012 (cit. a p. 14).

[28] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Berlin: Springer, 2006 (cit. a p. 15).

[29] John R. Searle. «Minds, brains, and programs». In: *Behavioral and Brain Sciences* 3 (1980), pp. 417–457 (cit. a p. 16).

[30] Warren S. McCulloch e Walter Pitts. «A logical calculus of the ideas immanent in nervous activity». In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133 (cit. a p. 17).

[31] Marvin Minsky e Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, Massachusetts: M.I.T. Press, 1969 (cit. a p. 18).

[32] J. M. Bourinet. «Reliability analysis and optimal design under uncertainty - Focus on adaptive surrogate-based approaches». Habilitation à diriger des recherches. Université Clermont Auvergne, 2018. URL: https://tel.archives-ouvertes.fr/tel-01737299 (cit. a p. 22).

[33] Bruno Sudret, Stefano Marelli e Joe Wiart. «Surrogate models for uncertainty quantification: An overview». In: *2017 11th European Conference on Antennas and Propagation (EUCAP)*. 2017, pp. 793–797 (cit. a p. 22).

[34] Bobak Shahriari, Kevin Swersky, Ziyu Wang e et al. «Taking the Human Out of the Loop: A Review of Bayesian Optimization». In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175 (cit. a p. 22).

[35] K. T. Fang, R. Li e A. Sudjianto. *Design and Modeling for Computer Experiments*. Chapman & Hall/CRC, 2005 (cit. a p. 23).

[36] I.S. Stievano, I.A. Maio e F.G. Canavero. «On-the-fly estimation of IC macromodels». In: *IEE Electronics Letters* (lug. 2006) (cit. a p. 36).