POLITECNICO DI TORINO
Facoltà di Ingegneria
Corso di laurea in Ingegneria Elettronica

Tesi di laurea

Tecniche simboliche per l'analisi di Macchine a Stati Finiti nell'ambito della verifica formale di correttezza di circuiti

Relatore: Dott.Gianpiero Cabodi
Candidato: Luca Galli

Dicembre 1997

Introduzione

Nella progettazione dei circuiti digitali ha assunto grande importanza la verifica di correttezza del progetto. La verifica può essere effettuata in diversi modi. Il primo metodo è la *simulazione* del circuito. La simulazione può essere adatta per circuiti di piccole dimensioni e aiuta ad aumentare la confidenza col circuito; è tuttavia inadeguata per circuiti di grosse dimensioni dove determinare per simulazione ogni possibile mal funzionamento diventa altamente improbabile (oppure può diventare troppo costoso in termini di consumo di tempo).
Nasce così l'esigenza di poter disporre di metodi che garantiscano la correttezza del circuito in ogni possibile situazione. Tali metodi vengono indicati con il termine di metodi di *verifica formale* in contrapposizione alla verifica informale ovvero la simulazione.

La verifica formale consiste nel disporre di un modello matematico col quale rappresentare il sistema, un linguaggio col quale esprimere le proprietà da verificare e, infine, un metodo di dimostrazione per verificare che il modello soddisfa le proprietà desiderate.
Uno dei paradigmi più importanti per la verifica formale è il *model checking* dove la proprietà da verificare è espressa in una logica opportuna denominata CTL (*Computation Tree Logic*) mentre il sistema è rappresentato da una FSM (Finite State Machine). Un esempio di FSM è rappresentato in figura 1.
In questo ambito, gioca un ruolo centrale l'attraversamento dello spazio degli stati della macchina a stati finiti.

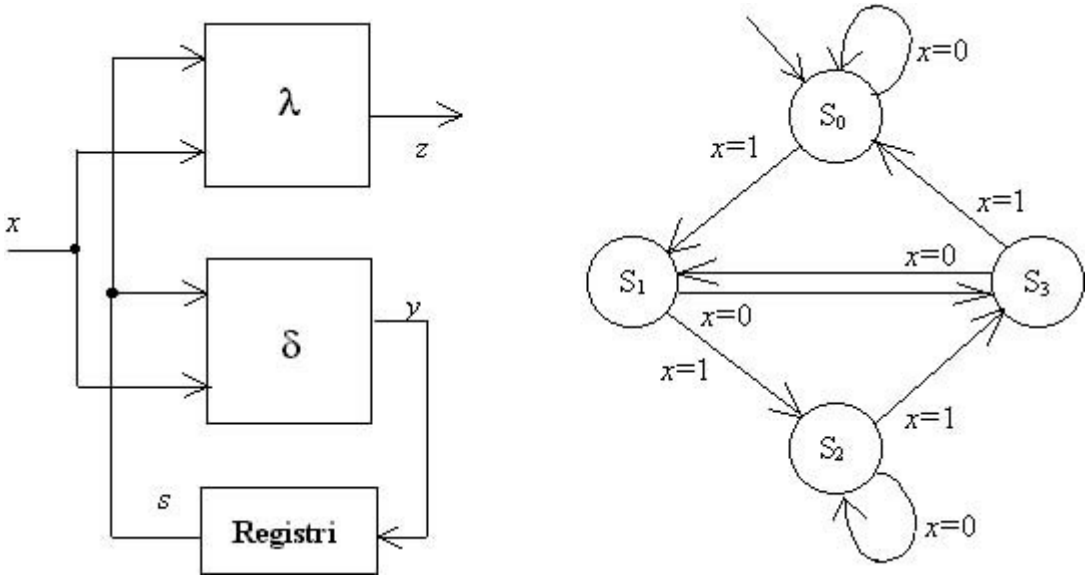

Figura 1: Macchina a Stati Finiti: schema strutturale (a sinistra) e diagramma stato - transizione (a destra)

La tesi di laurea si articola in due parti principali. La prima parte analizza l'aspetto teorico riguardante le

tecniche simboliche di attraversamento delle Macchine a Stati Finiti e della verifica formale di correttezza dei circuiti. La seconda parte è applicativa e riguarda la realizzazione di un pacchetto software denominato **PdTrav** (acronimo di *Politecnico di Torino Traversal*) che è in grado di attraversare macchine a stati finiti con i metodi esposti nella parte teorica.

Tale software, sviluppato sia per il sistema operativo UNIX che per il sistema operativo Windows 95, è stato realizzato dall'ing. Giampiero Cabodi (co-relatore di questa tesi), dall'ing. Stefano Quer del dipartimento di Automatica e Informatica del Politecnico di Torino, dal sottoscritto Luca Galli e dal tesista Fabio Maino.

I capitoli principali in cui è stata suddivisa la tesi di laurea sono sei; sono preceduti da un capitolo introduttivo e sono terminati da tre brevi appendici.

Il **capitolo 2** affronta lo studio teorico dei BDD (Binary Decision Diagram). I BDD permettono di rappresentare funzioni booleane in modo canonico, compatto ed efficiente. Grazie all'introduzione dei BDD, gli algoritmi di attraversamento e di verifica formale hanno raggiunto un alto grado di efficienza e, in particolare, è possibile analizzare circuiti digitali di grande complessità. Un esempio di BDD è riportato in figura 2.
Un paragrafo introduttivo esporrà i vari metodi per rappresentare le funzioni booleane (i metodi più conosciuti sono la tavola di verità e la mappa di Karnough), verrà poi presentato il metodo per costruire i BDD e le tecniche per una gestione efficiente al calcolatore. Verranno infine elencate alcune varianti dei BDD.

Il **capitolo 3** studia le tecniche di attraversamento delle macchine a stati finiti.
Vengono esposti i concetti fondamentali quali la definizione di macchina a stati finiti (FSM), di *transition relation* (TR) di una FSM, di *macchina prodotto*, di *immagine* di una funzione booleana. Infine verranno analizzati i vari metodi di attraversamento e di calcolo di immagine. Con particolare riguardo verrà descritto il metodo proposto da Rajeev Ranjan et al. [Ran95]. Tutte le tecniche descritte verranno poi realizzate in pratica dal software PdTrav.

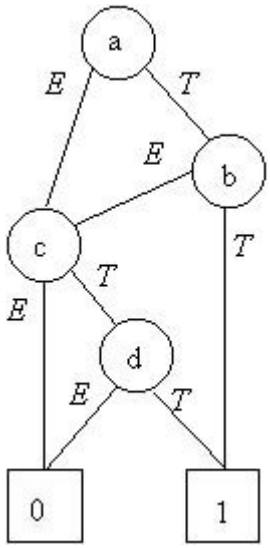

Figura 2: Esempio di BDD

Il **capitolo 4** analizza le tecniche di verifica formale dei circuiti digitali. Particolare enfasi verrà posta nella descrizione del paradigma del *model checking*, una tecnica di verifica formale particolarmente automatizzata e flessibile.
La tecnica consiste nel rappresentare il sistema ../mediante una FSM mentre le proprietà che devono essere verificate sono espresse facendo uso di una particolare logica temporale chiamata CTL (Computation Tree Logic). Essa permette di esprimere attraverso formule concise e non ambigue concetti espressi nel linguaggio naturale quali "qualche volta nel futuro ...", "è sempre vero che ...", "è vero fintantoché...".
L'ultimo paragrafo è dedicato alle condizioni di *fairness constrain* applicabili alle formule CTL.

Il **capitolo 5** studia in modo approfondito il pacchetto software **VIS** (Verification Interacting with Syntesis), sviluppato presso l'Università della California a Berkeley in collaborazione con l'Università del Colorado a Boulder.
VIS è in grado di fare la verifica formale, la simulazione e la sintesi di FSM. Per quanto riguarda la verifica formale, VIS attualmente accetta soltanto il paradigma del model checking. Verrà analizzato l'ambiente VIS, accessibile attraverso una semplice interfaccia testuale e verranno descritti sommariamente i linguaggi di

descrizione dell'hardware che VIS è in grado di leggere ovvero il linguaggio BLIF (Berkeley Logic Interchange Format), il linguaggio BLIF-MV (BLIF - Multi Valued) e il Verilog.
Verranno infine presentati i risultati di alcune prove effettuate in VIS.

Il **capitolo 6** analizza in modo dettagliato il pacchetto software **CUDD** (Colorado University Decision Diagram) sviluppato presso l'Università del Colorado a Boulder.
A differenza del pacchetto VIS, il pacchetto CUDD non possiede una interfaccia utente (se si esclude il sotto pacchetto NanoTrav) ma è una raccolta di funzioni che permettono di creare e manipolare in modo efficiente i BDD. Per questa ragione, PdTrav utilizza il pacchetto CUDD come libreria di funzioni di basso livello.
Verranno analizzate le strutture alla base della gestione dei BDD, mentre le principali funzioni del pacchetto verranno spiegate facendo ricorso a semplici esempi di utilizzo.

Il **capitolo 7** descrive il software **PdTrav**. Il programma verrà analizzato sotto due punti di vista differenti. Per primo, verrà descritto il funzionamento dal punto di vista dell'interfaccia utente. Tutte le funzioni e i dettagli implementativi sono nascosti ed il loro controllo avviene attraverso le *opzioni* che possono essere specificate. Questa prima parte si può considerare come il *manuale dell'utente*.
Successivamente si entrerà in dettaglio ad analizzare le principali funzioni realizzate. Questa seconda parte si può considerare come il *manuale del programmatore*.

Il pacchetto Pdtrav è stato suddiviso in cinque sotto pacchetti: DDI, FSM, TR, TRAV, PDUTIL.
DDI (Decision Diagram Interface) è il pacchetto di base. Esso permette di gestire cinque diverse entità: funzioni booleane, vettori di funzioni booleane, variabili, insiemi di variabili, vettori di variabili. Tutte le chiamate alle funzioni di CUDD passano per DDI.
Il pacchetto FSM legge la descrizione della FSM da file; TR contiene algoritmi per la gestione della transition relation e del calcolo di immagine, TRAV è il pacchetto principale che fornisce l'interfaccia utente e contiene la procedura di attraversamento principale; infine PDUTIL è una raccolta eterogenea di funzioni di utilità comuni a tutti i pacchetti.