Metodi formali per la verifica dei sistemi digitali

M. Favalli

Department of Engineering, Univ. of Ferrara

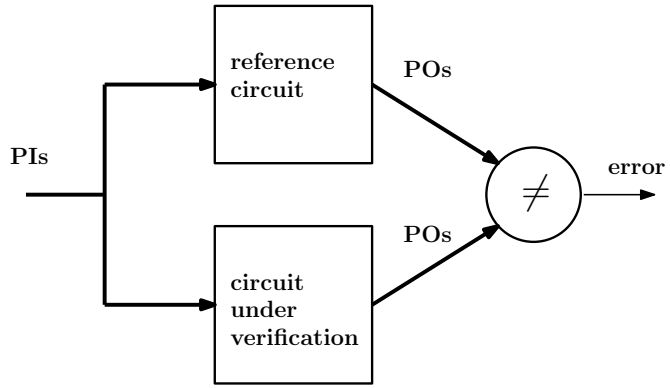
Sommario

Verifica dei sistemi digitali

- La complessitá degli attuali sistemi digitali richiede l'utilizzo di diversi livelli di astrazione nella loro descrizione
- La loro sintesi si muove da quelli piú alti (specifica) verso quelli piú bassi (implementazione)
- A ogni passo é necessario verificare che la rete prodotta dalla sintesi soddisfi le specifiche ai livelli superiori
- La verifica si occupa poi anche di svolgere ulteriori analisi (timing, fault tolerance) che riguardano i maggiori dettagli che si rendono disponibili durante la sintesi

Miter

Schema generale per la verifica (e il collaudo) dei circuiti digitali



Simulazione e metodi formali

- La simulazione costituisce il primo strumento di verifica ad essere stato sviluppato
  - vantaggi: applicabilitá, semplicitá di utilizzo
  - svantaggi: qualitá dei risultati nel caso di sistemi con un numero elevato di stati che non possono essere simulati in maniera esaustiva
- I metodi formali sono in grado di provare l'equivalenza (o meno) fra specifica e implementazione nello stesso modo in cui viene provato un teorema
  - vantaggi: qualitá dei risultati
  - svantaggi: incapacitá di fornire risposte parziali se la complessitá del sistema eccede le loro capacitá
- Nella pratica si tratta di soluzioni complementari

Esempi

- I metodi formali di verifica verranno instanziati tramite alcuni semplici esempi di reti combinatorie e reti sequenziali sincrone
- I metodi formali che utilizzeremo sono basati sulla Boolean Satisfiability
- La presenza di SAT solver particolarmente efficienti rende tale approccio il piú potente attualmente disponibile
- Si fornisce una breve introduzione alla Boolean Satisfiability

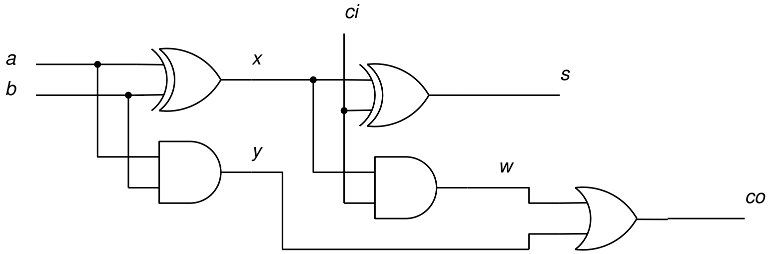
Modelli e solver

- I metodi formali si basano su modelli matematici del sistema da verificare e delle relative specifiche
- Alcuni esempi
  - Binary Decision Diagrams (BDDs)
    - modello funzionale
    - non sono ben applicabili ad alcuni tipi di circuiti
  - Boolean Satisfiability (SAT)
    - modello funzionale
    - sfruttano solver molto potenti
    - usati con AND-OR-INVERT Graphs (AIGs) recuperano alcune informazioni strutturali
  - Satisfiability Modulo Theories
    - generalizzazione dei modelli SAT, servono per descrizioni ad alto livello

Boolean Satisfiability (SAT)

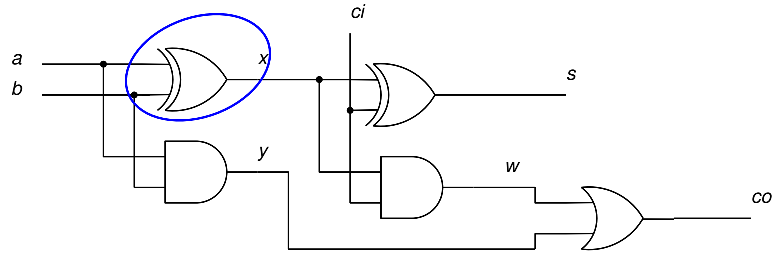
- Ciascun gate é descritto da formula CNF (PS) che ha un valore vero se gli assegnamenti ai uoi ingressi e alla sua uscita sono consistenti
- Esempio (OR gate): $\Phi_{OR} = o \leftrightarrow a + b$
  $\Phi_{OR} = (a \rightarrow o)(b \rightarrow o)(a'b' \rightarrow o')$
  $\Phi_{OR} = (o + a')(o + b')(o' + a + b)$ (CNF)
- La congiunzione (Φ) delle CNF di tutti i gate descrive tutti gli assegnamenti consistenti dei segnali di un circuito
- A tale CNF si possono aggiungere ulteriori vincoli per forzare alcuni segnali a determinati valori
  - nel miter si puó mettere $\Phi error = 1$

Esempio



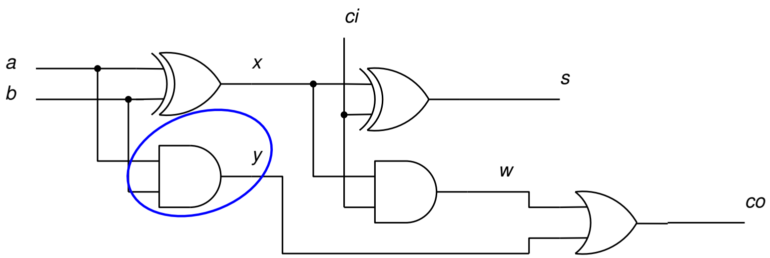
$$\Phi = (x'+a'+b')(x'+a+b)(x+a'+b)(x+a+b')(y'+a)(y'+b)(y+a'+b')$$
$$(s'+x'+ci')(s'+x+ci)(s+x'+ci)(s+x+ci')(w'+x)(w'+ci)(w+x'+ci')$$
$$(co'+y+w)(co+y')(co+w')$$

Esempio



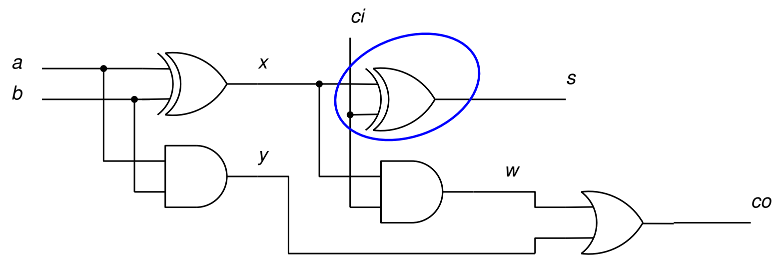
$$\Phi = (x'+a'+b')(x'+a+b)(x+a'+b)(x+a+b')(y'+a)(y'+b)(y+a'+b')$$
$$(s'+x'+ci')(s'+x+ci)(s+x'+ci)(s+x+ci')(w'+x)(w'+ci)(w+x'+ci')$$
$$(co'+y+w)(co+y')(co+w')$$

Esempio



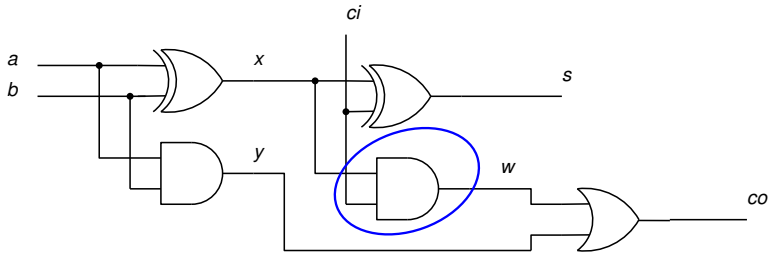
$$\Phi = (x'+a'+b')(x'+a+b)(x+a'+b)(x+a+b')(y'+a)(y'+b)(y+a'+b')$$
$$(s'+x'+ci')(s'+x+ci)(s+x'+ci)(s+x+ci')(w'+x)(w'+ci)(w+x'+ci')$$
$$(co'+y+w)(co+y')(co+w')$$

Esempio



$$\Phi = (x'+a'+b')(x'+a+b)(x+a'+b)(x+a+b')(y'+a)(y'+b)(y+a'+b')$$
$$(s'+x'+ci')(s'+x+ci)(s+x'+ci)(s+x+ci')(w'+x)(w'+ci)(w+x'+ci')$$
$$(co'+y+w)(co+y')(co+w')$$

Esempio



$$\Phi = (x'+a'+b')(x'+a+b)(x+a'+b)(x+a+b')(y'+a)(y'+b)(y+a'+b')$$
$$(s'+x'+ci')(s'+x+ci)(s+x'+ci)(s+x+ci')(w'+x)(w'+ci)(w+x'+ci')$$
$$(co'+y+w)(co+y')(co+w')$$

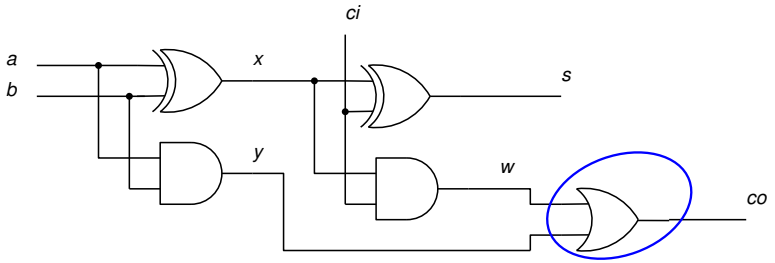
metodi formali

Introduzione

Boolean Satisfiability

Applicazioni

Esempio



$$\Phi = (x'+a'+b')(x'+a+b)(x+a'+b)(x+a+b')(y'+a)(y'+b)(y+a'+b')$$
$$(s'+x'+ci')(s'+x+ci)(s+x'+ci)(s+x+ci')(w'+x)(w'+ci)(w+x'+ci')$$
$$(co'+y+w)(co+y')(co+w')$$

metodi formali

Introduzione

Boolean Satisfiability

Applicazioni

SAT solvers

- Un SAT solver cerca un modello (i.e. un insieme di assignamenti alle variabili) che renda vera una formula CNF
- i SAT solver utilizzano ricerche basate sul branching
- alcune tecniche utilizzate da tali algoritmi:
  - unit clause propagation che permette semplificazioni a basso costo $((x_0 + x_1 + x_2 + ...)x_0 = x_0)$
  - analisi dei conflitti che producono nuove clausole che evitano di tornare ad esplorare sottospazi inutili
  - non-chronological backtracking: il solver torna alla prima decisione possibile causa di un conflitto
- Queste tecniche di learning rendono i SAT solver molto efficienti nel provare la non soddisfacibilitá

metodi formali

Introduzione

Boolean Satisfiability

Applicazioni

Strumenti utilizzati

- Si utilizzerá uno strumento di dominio pubblico per la sintesi e la verifica: **abc** (Berkeley)
- Prestazioni elevate ed algoritmi allo stato dell'arte
- Formati semplificati per la descrizione dei componenti
- La presentazione non discute gli aspetti di sintesi con cui sono stati ottenuti i componenti

Applicazioni

- Combinational Equivalence Checking
- Sequential Equivalence Checking
  - bounded
  - unbounded
- Property checking

Combinational Equivalence Checking (CEC)

- Verifica se due reti combinatorie sono equivalenti, ovvero se realizzano la stessa funzione
- Per ogni configurazione di ingresso le uscite devono avere lo stesso valore
- Con strumenti SAT é piú efficiente cercare di provare che esiste una configurazione di ingresso per cui almeno un uscita ha valore diverso nelle due reti
- Se non si riesce a trovarla, allora le reti sono equivalenti

Esempio di CEC

- Descrizione livello RTL di un Carry Saver Adder (CSA) che somma 3 interi rappresentati da 3 parole di 16 bit ciascuna
- Un circuito molto piccolo rispetto alle capacitá degli strumenti che stiamo considerando
- Nella sintesi, questo circuito RTL viene mappato su una libreria tecnologica
- Questo processo puó essere soggetto a errori e quindi si usa la verifica per provare l'equivalenza fra la descrizione RTL e la realizzazione a livello gate
- comando di **abc**: `cec`

Sequential Equivalence Checking

- Un circuito sequenziale sincrono puó essere trasformato in uno combinatorio tramite time-frame expansion
- Si possono quindi utilizzare gli stessi strumenti del CEC
- Il problema principale é chiaramente dato dal numero di time frame in cui espandere la rete sequenziale
- **Bounded Model Checking**: il numero di time frame viene fissato preventivamente
- **Unbounded Model Checking**: il numero di time frame viene aumentato dinamicamente fino a provare la non equivalenza o a esplorare lo spazio di possibili soluzioni provando l'equivalenza

Esempio di SEC

- Si utilizzano due contatori binari per illustrare le differenze fra bounded e unbounded model checking
- Boundend model checking: comando **`bmc`**
- Unbounded model checking: comando **`dprove`**

metodi formali

Introduzione

Boolean Satisfiability

Applicazioni

Conclusioni

- Tutti gli esempi considerati sono di dimensioni ridottissime
- I SAT solver possono arrivare a decine di migliaia di clausole e variabili
- Utilizzando strumenti commerciali i flussi di sintesi e verifica sono meglio integrati