Linguaggi di descrizione dell'hardware

Progetti a.a. 2018/19

Lista provvisoria e incompleta

I progetti vengono assegnati dal docente sulla base delle preferenze degli studenti. Si raccomanda di inserire come soggetto in qualsiasi mail riguardante i progetti la seguente stringa: `HDL - nome cognome`

Informazioni di carattere generale che possono essere utili in diversi progetti.
Ritardi realistici dei gate

In questo corso non si indirizzano aspetti specifici a una data tecnologia, ma comunque puó valere la pena mettere valori di ritardo compatibili con qualche tecnologia recente, per cui si consiglia di mettere questi valori (se servono).

NOT	$(30 + \text{fan-out} \cdot 15)\ ps$
NAND/NOR	$(40 + \text{fan-out} \cdot 20)\ ps$
AND/OR	$(50 + \text{fan-out} \cdot 20)\ ps$
XOR	$(50 + \text{fan-out} \cdot 30)\ ps$

Modello VHDL di componente che genera sequenze pseudocasuali di vettori logici

```
entity lfsr is
  generic(n: natural);
  port(reset,clk: in std_logic;
       q: out std-logic_vector(0 o n-1));
end entity lfsr;

architecture behav of lfsr is
begin
  process (clk)
  constant initial:="01011101....";
  variable tmp: std_logic_vector(0 to n-1);
  variable bk: std_logic;
  -- any value different from all 0s
  begin
    if (rising_edge(clk)) then
      if (reset='0') then
        bk:=tmp(1) xor tmp(n-1);
        for i in 1 to n-1 loop
          tmp(i):=tmp(i-1);
        tmp(0):=b;
      else
        tmp:=(others=>'0');
      end if;
    end if;
    q <= tmp;
  end process;
end architecture;
```

Progetto 1

Modello behavioral dei ritardi di un adder

Data una descrizione strutturale di un n-bit adder (ripple-carry), si vuole costruire un modello comportamentale dei ritardi fra ingresso e uscita.

Le operazioni da svolgere sono le seguenti:

1. costruire un modello strutturale del componente (schema logico)
2. eseguire la STA su tale modello determinando EAT e LST per ciascuna uscita
3. costruire una prima versione di modello VHDL in cui il cambiamento delle uscite avviene rispettando EAT e LST
   (a) se un uscita cambia in conseguenza di un cambiamento degli ingressi, l'uscita si porta prima a 'X' con ritardo EAT e poi al suo valore finale con ritardo LST
4. il modello ha la caratteristica rilevante di essere conservativo su ritardo minimo e massimo
5. il componente va simulato in parallelo alla sua realizzazione a livello gate valutando cosí l'errore medio
6. é poi interessante provare a connettere in cascata un paio di questi adder

Ci sono diverse versioni alternative di questo problema che tengono conto degli hazard o cercano di ridurre gli errori sui ritardi. Ad esempio é possibile calcolare EAT e LST rispetto a ciascun ingresso e poi da questo dato calcolare i tempi di ritardo delle uscite come il minimo e il massimo degli EAT e LST calcolati rispetto agli ingressi che commutano.

Progetto 2

Guasti nella rete di distribuzione del clock

Si vogliono valutare gli effetti di possibili guasti della rete di distribuzione del clock.

Serve introdurre un po di terminologia, si chiama clock source la sorgente di tale segnale supposto ideale e clock sinks gli ingressi di tipo clock dei FF. Per diversi motivi (difetti e disturbi) il segnale di clock puó non arrivare in maniera corretta a uno piú FF.

Si noti che la rete di distribuzione di clock presenta inevitabilmente degli skew, che peró qui non verranno tenuti in conto perché facciamo l'ipotesi che siano giá stati tenuti in conto durante il progetto. Quindi supporremo di avere una rete che in assenza di disturbi ha un funzionamento ideale (il clock arriva in maniera sincrona a tutti i sink).

In questo progetto si vuole modellare in VHDL questo tipo di problemi nel caso in cui siano dovuti a disturbi (ad esempio crosstalk). In particolare, si fa l'ipotesi che il problema riguardi un singolo FF e che al suo sink si presenti uno dei seguenti problemi:

- il clock ritarda rispetto a quello ideale
- il clock anticipa rispetto a quello ideale
- il clock presenta un fronte di salita indesiderato oltre a quello normale (jitter)

tali difetti sono parametrici, ovvero il loro effetto dipende dal valore di ritardo/anticipo che presentano. L'effetto di questi guasti va modellato in VHDL e analizzato tramite la simulazione di descrizioni strutturali di FSM. I guasti possono essere modellati sia inserendo opportuni buffer nella rete di distribuzione del clock o modificando il modello VHDL dei FF.

Progetto 3

Detection di timing violation

Come é noto, nel caso di violazioni del timing, un errore puó essere prodotto e venire campionato dagli elementi di memoria. Per rivelare questi errori, occorre simulare il circuito e confrontare i valori campionati con quelli corretti. Si tratta quindi di costruire un miter per il circuito in esame. In questo progetto, si propone un alternativa pessimistica per tracciare questi errori. Si parte dall'ipotesi che il circuito sia sincrono, a ciclo singolo e che campioni agli istanti kT $k = 0....n$. In questo caso, si puó avere una timing violation se e solo se esiste almeno un gate nel circuito che viene valutato nell'intervallo $[(k-1)T, kT[$ e produce la sua risposta a un tempo $t > kT$.

Quindi se entro nel processo che calcola l'uscita del gate al tempo t (notate che si puó usare la funzione `now`) e l'uscita viene aggiornata a $t + del$, allora si ha un potenziale errore se $t \ mod \ T \neq (t + del) \ mod \ T$.

Si noti che in questo caso ci si accorge di errori senza bisogno di propagarne gli effetti alle uscite del circuito.

Nel progetto si richiede quindi di creare una libreria di gate logici standard con anche il periodo di clock passato come generic. Ogni gate deve essere in grado di valutare la condizione precedente producendo un indicazione di possibile erorre con una assert. Questa libreria va provata per diversi tipi di rete (ripple adder, CLA adder).

Progetto 4

In questo progetto si deve realizzare un programma in un qualsiasi linguaggio (C, Java etc. etc.) che legge una descrizione dataflow di una una qualsiasi rete combinatoria, determina la lista dei segnali di tale rete (uscite e segnali interni) e realizza una versione modificata del componente e un testbench che siano in grado di eseguire una simulazione di guasti di tipo stuck-at sotto l'ipotesi di guasto singolo. Questa implementazione deve utilizzare le istruzioni messe a disposizione dallo standard VHDL 2008 (`force` , `release`).

In particolare, si tratta di:

- costruire un modello VHDL del circuito in cui si aggiunge un segnale intero di ingresso x e un segnale di tipo `std_logic` t. Il segnale x corrisponde all'indice del segnale che deve essere bloccato al valore specificato da$t$ (se $x = 0$ nessun segnale é guasto)

- il nuovo modello aggiunge alle istruzioni dataflow precedenti un processo con x e t nella sensitivity list che sulla base del valore di x utilizzando la `force` blocca a t un segnale fino a quando non cambiano tali parametri e allora con la `release` ne ripristina il funzionamento corretto per poi passare al guasto successivo

- il testbench istanzia la rete corretta e quella col guasto poi ha un processo che cambia il valore x (inizialmente a 0), si sospende attivando un ulteriore processo che genera un insieme di stimoli applicati alle due copie del circuito e si sospende fino a quando tale processo non termina

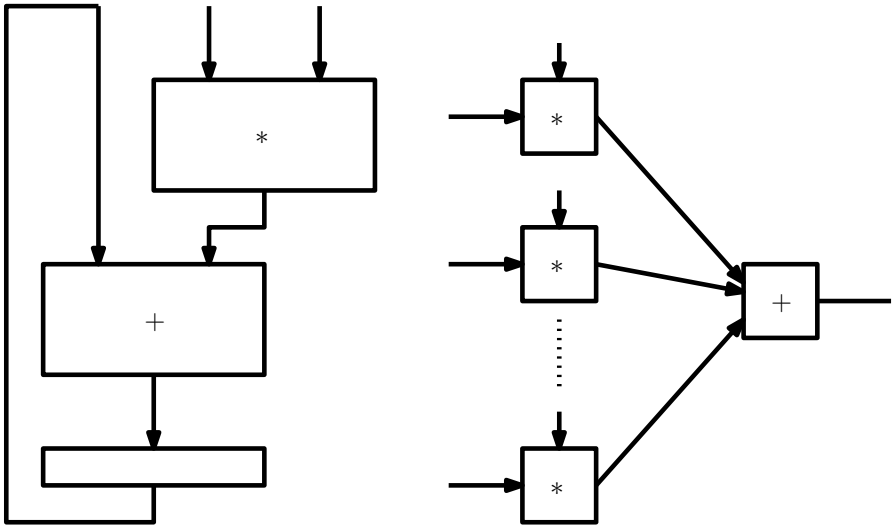
- serve anche una rete che confronti le uscite dei due circuiti determinando se il guasto viene rivelato o meno come un errore

- alla fine si dovrebbe avere la percentuale (*fault coverage*) di guasti rivelati dalla sequenza in esame

Progetto 7

Confronto fra MAC e architettura combinatoria

Una parte importante delle reti neurali esegue il calcolo di una somma pesata $\sum_i x_i w_i$, dipendentemente dalla topologia della rete e da alcuni aspetti che riguardano throughput e consumo di potenza, questa puó essere fatta tramite una rete Multiply and Accumulate (MAC, a sinistra) o tramite un implementazione combinatoria (a destra). Dal punto di vista funzionale le due architetture sono completamente equivalenti.



Non é cosí dal punto di vista dei guasti, ad esempio, un guasto nel moltiplicatore del MAC disturba tutti i prodotti, mentre nel caso di guasto in uno dei moltiplicatori della rete combinatoria, il guasto disturba un solo prodotto.

In questo progetto si tratta di valutare quantitativamente questa differenza fra i due errori sul risultato finale. I passi da realizzare sono i seguenti:

- realizzazione strutturale delle due architetture (per le reti si vedano le dispense di ASCD)
- realizzazione di un test bench per ciascuna di esse (il test bench deve contenere anche una valutazione behavioral del risultato in assenza di guasti)
- iniezione di un certo numero di guasti nelle due architetture e valutazione dell'errore (i guasti, perlomeno nei moltiplicatori, dovrebbero essere iniettati in posizioni omologhe nei due circuiti);

Progetto 8

Interfaccia per l'inserimento di descrizioni strutturali in VHDL

In questo progetto (per due persone) si vuole costruire un sistema in grado di facilitare la costruzione di modelli VHDL strutturali. Si tratta di provare alcuni concetti senza sviluppare un sistema completo. Il progetto richiede l'utilizzo un interfaccia con finestre di testo, forms, bottoni .... ma non grafica vera e propria. Gli strumenti di programmazione possono essere i piú diversi, da Java a C con librerie tipo Qt, python, tcl/tk ....

In pratica, il sistema deve consentire l'inserimento di componenti e la possibilit di richiamarli in fase di instanziazione, il mantenimento di una lista di segnali disponibili associabili alle porte dei componenti e la possibilitá di salvare i risultati in un formato intermedio. Devono essere messe poi a disposizione utiliá che svolgano funzioni piú o meno utili che non sono messe a disposizione dal VHDL.

Ad esempio, dato un segnale il sistema deve fornire la lista dei driver e il suo fan-out, oppure dato un ingresso il sistema deve fornire i cammini da quell'ingresso alle uscite. Ed eventuali altre funzioni ritenute utili dagli autori del progetto.

Esempio di possibile layout

Componenti	Segnali	Istanze
and2	x	a0: or2 port map(x,y,w);
or2	y	a1: and2 port map(x,y,w);
....	w	a2: and2 port map(x,w,z);
....	z	....
working area with buttons and windows		

In presenza di specifiche competenze, il progetto puó essere svolto anche utilizzando un interfaccia web (js, php ....).

Progetto 10

Simulazione Montecarlo di errori timing

Errori timing

Simulazione dello stesso circuito sincrono con diversi valori di ritardo dei gate secondo una distribuzione uniforme di probabilitá dei ritardi dei gate.

Analisi della dipendenza dell'errore dal periodo di clock e dalla configurazione di ingresso applicata. Bisogna applicare un insieme sufficentemente grande di vettori di ingresso e simulare con tali stimoli lo stesso circuito per diverse configurazioni dei ritardi. Problema: come determinare se l'uscita campionata corretta?

Gestione del non-determinismo

- Supporto per alcuni dei progetti precedenti
- Libreria per la generazione di numeri casuali
- Realizzazione di un gate con possibilit di:
  - Ritardo variabile
  - Iniezione di errori transitori
- La stessa libreria si pu utilizzare per realizzare test bench con generazione pseudocasuale di vettori di collaudo
- Il package di base e tratto da un libro di C. Myers sulle reti asincrone

```
library ieee;
use ieee.math_real.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

package nondeterminism is

  shared variable s1:integer:=844396720;
  shared variable s2:integer:=821616997;

  -- Returns a number between 1 and num.
  impure function selection(constant num:in integer) return integer;
  -- Returns a std_logic_vector of size bits between 1 and num.
  impure function selection(constant num:in integer;
                            constant size:in integer) return std_logic_vector;
  -- Returns random delay between lower and upper. (m.u. ps)
  impure function delay(constant l:in integer;
```

```
                        constant u:in integer) return time;

end nondeterminism;

package body nondeterminism is ..

library ieee;
use ieee.std_logic_1164.all;
use work.nondeterminism.all;

entity inverter is
generic (index: integer; -- index of the element
         low_del,high_del: integer); -- low and high bounds on propagation delay
port(a: in std_logic;
     b: out std_logic;
     set: in boolean); -- signal provoking a new evaluation of delays
end entity inverter;

architecture behav of inverter is
signal indx: integer:=index;
begin
  process(a,set)
  variable del: time:=0 ps;

begin
  -- at the beginning of a new simulation sets a
  -- random quantity for propagation delay
  if (set'event) then
    del:=delay(low_del,high_del);
  end if;
  b <= not a after del;

 end process;
end architecture behav;
```

Progetto 11

Realizzazione di un simulatore logico event-driven

In questo progetto, si vuole realizzare un simulatore logico di tipo event-driven utilizzando un qualsiasi linguaggio ad alto livello (C, Java). Il simulatore legge una descrizione di una rete al livello gate con ritardi e un insieme di configurazioni di ingresso per tale rete e produce una previsione sul comportamento dei segnali della rete in risposta a tali stimoli.

Il formato della descrizione potrebbe essere una restrizione del dataflow del VHDL:

```
p <= a nand b after 2 ns;
q <= c and p after 1 ns;
....
```

Il programma legge tale descrizione e memorizza le informazioni in una tabella che ha un record per ogni nodo contenente le informazioni sul tipo di funzione, gli operandi e il ritardo. C'é poi una struttura dati che si utilizza per la simulazione che é essenzialmente una lista gerarchica con un elemento per ogni istante in cui la rete é attiva e tale elemento punta a una lista contenente i segnali da aggiornare in tale istante. L'algoritmo di simulazione inizializza tale struttura dati con gli eventi sugli ingressi e poi itera cambiando di valore ai segnali nella lista corrente e poi passa a valutare i gate nel fan-out di tali segnali programmando in istanti futuri gli eventi in uscita a tali gate. Questa metodologia é descritta anche sul testo.

Progetto 12

STA in VHDL

In questo progetto, si vuole realizzare un algoritmo che esegua la static timing analysis utilizzando un qualsiasi linguaggio ad alto livello (C, Java). Il simulatore legge una descrizione di una rete combinatoria al livello gate con ritardi.

Il formato della descrizione potrebbe essere una restrizione del dataflow del VHDL:

```
library ieee; -- ignored
use ieee.std_logic_1164.all; -- ignored

p <= a nand b after 2 ns;

q <= c and p after 1 ns;

....
```

Il programma legge tale descrizione e memorizza le informazioni in una tabella che ha un record per ogni nodo contenente le informazioni sul tipo di funzione, gli operandi e il ritardo. Utilizzando questa tabella si puó poi realizzare l'algoritmo di STA e produrre anche la lista di tutti i possibili cammini ingresso-uscita con i relativi ritardi.

Progetto 14

Stochastic computing

Si tratta di un alternativa ai sistemi digitali classici (che peró utilizza semplici porte logiche e flip-flop).

L'informazione é portata da una probabilitá di segnale (la probabilitá di avere un 1) di sequenze pseudocasuali generate nel circuito.

Si supponga di avere due sequenze pseudocasuali (sincrone) non correlate aventi rispettivamente $P(1) = P_x$ e $P(1) = P_y$, se queste sequenze entrano in un gate AND, la probabilitá di avere 1 in uscita é $P_x P_y$ e quindi si é realizzato un moltiplicatore (reali ¡ 1) con un solo AND.

Esempio: $p_x = 0.5$, $p_y = 0.6$ da luogo in questo caso a $p_w = 0.25$ (invece del valore esatto). L'accuratezza del metodo dipende dalla lunghezza delle sequenze che devono essere fra loro non correlate.

```
x 01101010100101100110
y 11010101010111001110
w 01000000000101000110
```

La somma invece da luogo problemi. Si tratta di realizzare in VHDL una libreria di componenti (LFSR, decorrelatori ....) per questo tipo di applicazioni. Verrá fornito un articolo con gli schemi dei circuiti da implementare. Le simulazioni devono valutare la precisione del metodo considerato in funzione di alcuni parametri di progetto.

Da spiegare un po meglio

Progetto 15

Approximate computing

L'approximate computing consiste nel consentire all'hardware di effettuare calcoli di tipo aritmetico con una precisione variabile. Dipendentemente dall'applicazione e dalle condizioni al contorno (energia disponibile), la rete puó decidere di approssimare in vari modi i bit meno significativi del risultato, consentendo di mantenere non attive le porte logiche che in condizioni normali dovrebbero calcolarle. Questo chiaramente consente di risparmiare potenza dinamica.

In questo progetto c' da implementare un adder in grado di calcolare una somma in maniera approssimata sulla base di un segnale di controllo. L'idea é quella di utilizzare solo un numero limitato di bit degli operandi in ingresso. L'adder é descritto in un articolo piuttosto semplice.

Progetto 16

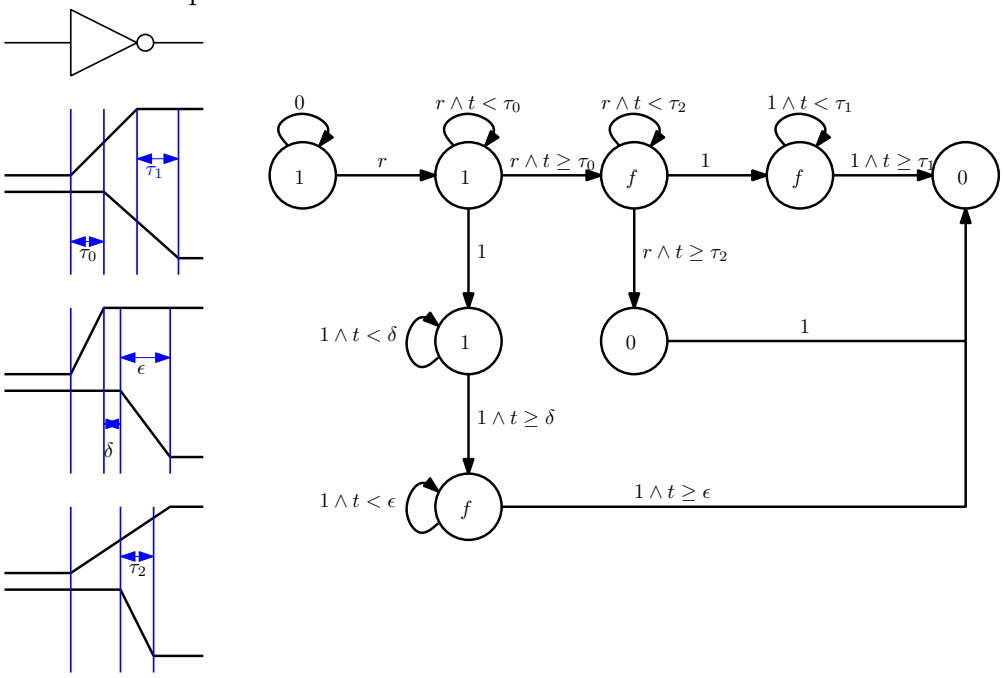
Caratterizzazione di un invertitore come timed automata

Il modello di ritardo di propagazione non tiene conto della pendenza dei segnali di ingresso e uscita. Questo puó essere verificato con simulazioni al livello elettrico.

Questi aspetti possono essere modellati in VHDL utilizzando un estensione delle FSM detta timed automata che nel caso piú semplice rappresenta un sistema in cui le transizioni avvengono a causa di cambiamenti degli ingressi o del passare del tempo.

In pratica, quando si entra in uno stato viene messa a 0 una variabile t che poi avanza col tempo di simulazione fino a quando una condizione su una transizione non diventa vera.

Un esempio di parte di un timed automata in grado di gestire tre diverse possibili condizioni sulle pendenze di ingressi e uscite di un invertitore é illustrato nella figura seguente. Tale automa va completato utilizzando simulazioni con SPICE per determinare i parametri.



Progetto 18
Floating point

Progetto 19
PSL