Nome _____Cognome _____

Compito di linguaggi di descrizione dell'hardware

Esercizio 1 Si descriva al livello comportamentale del VHDL un componente asincrono che ha il compito di contare i fronti di salita che si presentano sull'ingresso p. Devono essere contati solo i fronti relativi a impulsi (010) la cui ampiezza risulta $< \tau$, ove τ é passato come generic. Oltre all'ingresso su cui arrivano gli impulsi, supponiamo che sia presente anche un segnale di reset r. Il conteggio é codificato in binario sulle 4 uscite $s_{3..0}$ (il conteggio avviene in modulo 16). Quando $r = 1$ il conteggio si resetta producendo le uscite a 0 (pt. 5.0).

Soluzione

Si propone la soluzione concettualmente piú semplice, L'idea é di avere due copie del segnale p ritardate di t, una inerziale e una col ritardo di tipo trasporto. Quella inerziale non ha gli impulsi piccoli. Poi le uso per incrementare due contatori quello che viene incrementato dal segnale "trasporto" conta tutti gli impulsi e quello dal segnale "inerziale" conta solo quelli grandi. La loro differenza da quindi gli impulsi piccoli.

```
-- Code your design here
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;

entity count is
  generic(t: time);
  port(p,r: in std_logic;
       s: out std_logic_vector(3 downto 0));
end entity;

architecture behav of count is
signal pi,pt: std_logic;
begin

  pi <= p after t;
  pt <= transport p after t;

  process(pi,pt,r)
  variable tmpi,tmpt,tmp: unsigned(4 downto 0);
  begin
```

```
  if (r='1') then
    tmpi:="00000";
    tmpt:="00000";
    tmp:="00000";
  elsif (r='0') then
       if (rising_edge(pi)) then
         tmpi:=tmpi+"00001";
       end if;
       if (rising_edge(pt)) then
         tmpt:=tmpt+"00001";
       end if;
       tmp:=tmpt-tmpi;
   end if;
   s <= std_logic_vector(tmp(3 downto 0));
   end process;
end architecture;
```

Esercizio 2 Si descriva al livello comportamentale del VHDL un componente combinatorio che riceve in ingresso due parole a e b da 8 bit ciascuna. Tali ingressi codificano due interi con segno rappresentati in complemento a 2, A e B, rispettivamente. La rete deve produrre sull' uscita w anch'essa codificata nello stesso modo degli ingressi la differenza $A - B$ se tale risultato é correttamente rappresentabile su 8 bit. Nel caso in cui il risultato sia maggiore del piú grande intero in complemento a 2, allora deve essere prodotto tale valore. Se, invece, il risultato é minore del piú piccolo intero rappresentato in quel modo, allora deve essere prodotto tale intero (pt. 5.0).

In complemento a 2 con n bit, il piú grande intero é $2^{n-1} - 1$ e il piú piccolo intero é -2^{n-1}.
Soluzione

```
library ieee;
use ieee.std_logic_1164.all,ieee.numeric_std.all;

entity arithm is
  port(a,b: in std_logic_vector(7 downto 0);
       output: out std_logic_vector(7 downto 0));
end entity arithm;

architecture behav of arithm is
begin
  process(a,b)
  variable atmp,btmp,res: signed(8 downto 0);
  begin
    atmp:=signed(a(7) & a); -- sign extension
    btmp:=signed(b(7) & b);
```

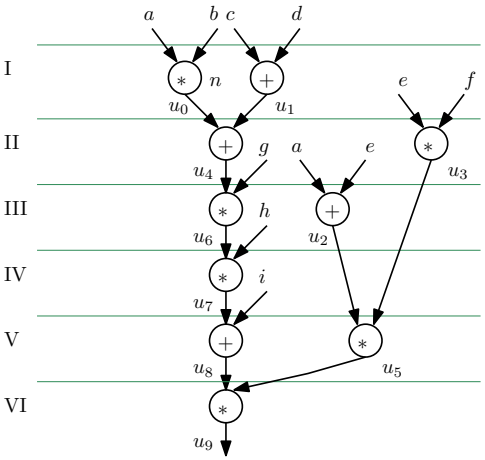
```
    res:=atmp-btmp;
    if (res<"110000000") then
      res:="110000000";
      elsif (res>"001111111") then
        res:="001111111";
    end if;
    output<=std_logic_vector(res(7 downto 0));
  end process;
end architecture;
```

Esercizio 3

Si consideri il seguente algoritmo:

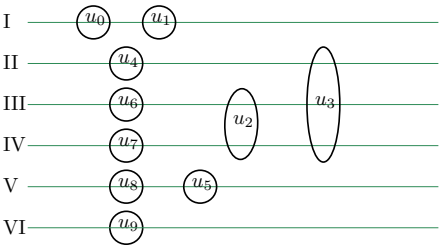
```
u0:=a*b;      u5:=u2*u3;
u1:=c+d;      u6:=u4*g;
u2:=a+e;      u7:=u6*h;
u3:=e*f;      u8:=u7+i;
u4:=u0+u1;    u9:=u8+u5;
```

si tracci il DFG. Si determini, utilizzando l'ipotesi di ciclo singolo, uno scheduling che, utilizzando un solo adder e un solo moltiplicatore, minimizzi la latenza. Si ottimizzi poi il numero dei registri utilizzati e si descrivano sinteticamente le operazioni al livello RTL (pt. 5.0).

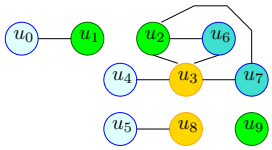


Binding e descrizione RTL

	adder	mult
I)	R3:=c+d;	R1:=a*b;
II)	R1:=R1+R3	R2:=e*f;
III)	R3:=a+e;	R4:=R1*g;
IV)		R4:=R4*i;
V)	R2:=R4+h;	R1:=R3*R2;
VI)	R3:=R2+R1;	



Tempo di vita delle variabili



Ottimizzazione del numero di registri tramite graph coloring