nome _____cognome _____matricola _____

Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi il modello comportamentale in VHDL di un componente che ha un ingresso *clk* e un uscita *pulse*. Il componente in presenza di un fronte (salita o discesa) su *clk* genera in uscita un impulso (010) la cui durata é definita da un parametro passato come generic.

Soluzione

Questa é la soluzion piú semplice, ne sono possibili diverse. Si noti che se si usa invece un processo, con due assegnamenti in sequenza, i secondo deve essere di tipo trasporto.

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

entity pulse_generator is
   generic(d: time);
   port(clk: in std_logic;
        pulse: out pulse);
end entity;
architecture behav0 of pulse_generator is
signal delayed: std_logic;
begin
  delayed<=transport clk after d; -- transport opzionale
  pulse<=clk xor delayed;
end architecture;
```

Esercizio 2

Si realizzi un modello comportamentale in VHDL di una rete sincrona che riceve in ingresso una parola di n bit $a_{n-1..0}$ (che rappresenta un numero intero positivo), un segnale di *reset* e un segnale di *enable*, oltre al segnale di *clock* (di cui deve essere considerato il fronte di salita). Compito della rete é sommare i valori di a che arrivano quando il segnale di *enable* é alto e produrre sull'uscita $s_{n-1..0}$ il risultato. Il segnale di *reset* riporta a 0 il risultato della somma.

a	-	4	4	7	1	10	6	2
reset	1	0	0	0	0	0	0	0
enable	-	1	1	0	1	0	0	1
s	0	4	8	8	9	9	9	11

Soluzione

Si nota che non vengono considerati i casi in cui i segnali di controllo sono a $'X'$.

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

entity sum_seq is
  generic(n: natural);
  port(a: in std_logic_vector(n-1 downto 0);
       reset,enable: in std_logic;
       clk: in std_logic;
       sum: out std_logic_vector(n-1 downto 0);
end entity;

architecture behav of sum_seq is
begin

  process(clk)
  variable tmp: unsigned(n-1 downto 0);
  begin
    if ((clk'last_value='0') and (clk='1')) then
      if (reset='1') then
        tmp := others(=>'0');
      elsif ((reset='0') and (enable='1')) then
        tmp := tmp + unsigned(a);
      end if;
      sum <= std_logic_vector(tmp);
    end if;
  end process;
end architecture;
```

Esercizio 3

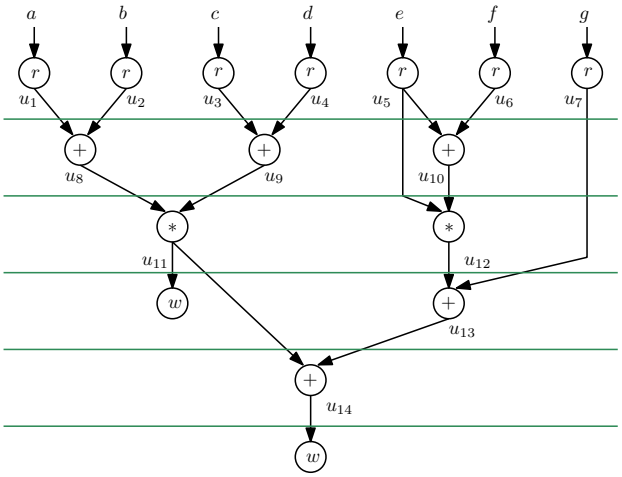
Si consideri il seguente algoritmo:

```
u1:=read a;
u2:=read b;
u3:=read c;
u4:=read d;
u5:=read e;
u6:=read f;
u7:=read g;
u8:=u1+u2;
u9:=u3+u4;
u10:=u5+u6;
u11:=u8*u9;
write u11;
u12:=u5*u10;
u13:=u12+u7;
u14:=u11+u13;
write u14;
```

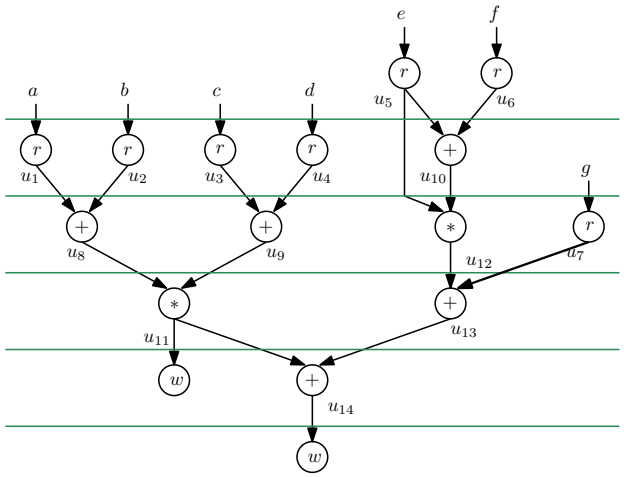
si tracci il DFG e si determinino lo scheduling ASAP e quello ALAP nell'ipotesi di ciclo singolo indicando latenza e risorse allocate per ciascuno scheduling. Si determini poi uno scheduling a risorse assegnate che minimizzi la latenza. Le risorse sono date da 1 moltiplicatore, 2 adder e 3 bus (bidirezionali) per gestire le operazioni di input/output.

Soluzione

Scheduling ASAP



Scheduling ALAP



Scheduling ottimizzato