Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi un modello comportamentale in VHDL di un componente che ha due ingressi d e res, e un uscita e. La rete riceve due parametri generic t_0 e t_1 di tipo time. In presenza di un impulso 010 di durata τ sull'ingresso d (mentre $res = 0$), l'uscita e si porta a 1 se $t_0 < \tau$, altrimenti rimane a 0. Quando res si porta a 1, l'uscita si porta a 0. Le transizioni di e avvengono con ritardo t_1 (pt. 5.0).

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

entity sens is
  generic (t0,t1: time);
  port(d,reset : in std_logic;
       e : out std_logic);
end entity sens;

architecture delay of sens is
signal dt: std_logic;
begin
  dt <= d after t0;

  process(d,reset)
  begin
    if (reset='1') then
      e <= '0' after t1;
    elsif (d'event) and (d='0') and (reset='0') then
      if (dt='1') then
        e <= '1' after t1;
      else
        e <= '0' after t1;
      end if;
    end if;
  end process;
end architecture;
```

Esercizio 2

Si realizzi la descrizione comportamentale di una rete combinatoria che riceve in ingresso due parole $a_{7..0}$ e $b_{7..0}$ che rappresentano due interi senza segno A e B. Compito della rete é produrre nell'uscita $o_{7..0}$ il valore di $2 * A + B/2$ (modulo 256) codificato in binario (pt. 5.0).

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

```
entity eval is
  port(a,b: in std_logic_vector(7 downto 0);
       o: out std_logic_vector(7 downto 0));
end entity;

architecture behav of eval is
begin
process(a,b)
variable at,bt: std_logic_vector(7 downto 0);
begin
  at:=a(6 downto 0) & '0';
  bt:='0' & b(7 downto 1);
  o<=std_logic_vector(unsigned(at)+unsigned(bt));
end process;
end architecture;
```

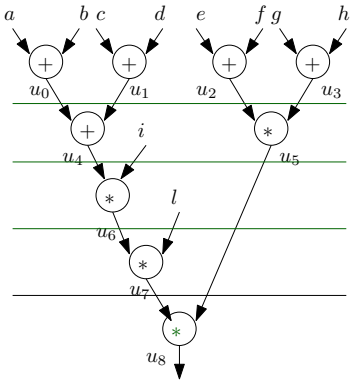
Esercizio 3

Si consideri il seguente algoritmo:

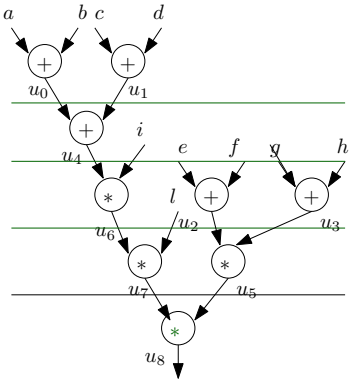
```
u0:=a+b;
u1:=c+d;
u2:=e+f;
u3:=g+h;
u4:=u0+u1;
u5:=u2*u3;

u6:=u4*i;
u7:=u6*l;
u8:=u7*u5;
```

si tracci il DFG e si determinino lo scheduling ASAP e quello ALAP nell'ipotesi di ciclo singolo. Si determini poi uno scheduling, che per una latenza pari a quella minima, ottimizzi le risorse nell'ipotesi $C_{add} \ll C_{mult}$ (ove C_{add} rappresenta il costo di un adder e C_{mult} quello di un moltiplicatore). Si minimizzi poi il numero di registri utilizzati per tale scheduling e infine si descrivano le operazioni svolte al livello RTL. Quale sarebbe invece lo scheduling ottimale se (per assurdo) $C_{add} \simeq C_{mult}$? (pt. 5.0).

Nella soluzione di questo esercizio riportiamo solo lo scheduling ASAP che é poi quello a costo minimo, e discutiamo brevemente il problema posto dall'ultimo quesito.



Si mostra lo scheduling ASAP che coincide anche con quello di costo minimo nell'ipotesi che il costo dell'adder sia di molto inferiore a quello del moltiplicatore



Lo scheduling ALAP invece diventa conveniente se il costo di questa realizzazione é inferiore a quello della preceddente. Ció consente di quantificare anche le ipotesi sul costo dei moltiplicatori e dei sommatori.

$$2C_{mult} + 2C_{add} < C_{mult} + 4C_{add}$$

$$C_{mult} < 2C_{add}$$

Quindi la seconda soluzione conviene se un moltiplicatore costa meno di due sommatori (poco realistico).