Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi un modello comportamentale in VHDL di un flip-flop che sia in grado di campionare sul fronte di salita del segnale di clock o su quello di discesa dipendentemente da un ingresso di controllo t (sul fronte di salita se $t = 0$ e su quello di discesa se $t = 1$). La rete riceve due parametri generic t_r e t_{setup} di tipo time che devono essere usati come tempi di risposta e di setup sia per il fronte di salita che per quello di discesa (pt. 5.0).

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

entity dff_rf is
  generic(tr,tsu: time);
  port(d,clk: in std_logic;
       t: in std_logic;
       q: out std_logic);
end entity dff;

architecture behav of dff_rf is
begin
  process(clk)
  begin
    if ((clk='1') and (clk'last_value='0') and (t='0')) then
      if (d'stable(tsu)) then
        q <= d after tr;
      else
        q <= 'X' after tr;
      end if;
    elsif ((clk='0') and (clk'last_value='1') and (t='1')) then
      if (d'stable(tsu)) then
        q <= d after tr;
      else
        q <= 'X' after tr;
      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 con segno A e B. Compito della rete é produrre nell'uscita $o_{7..0}$ il valore di $2 * A + B/2$ (modulo 128) 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;

-- in questa versione si cerca di ridurre
-- la probabilit  di overflow dimensionando at e quindi bt
-- in modo da non avere overflow quando si calcola 2*A
-- es. se A=-8 e B=5, se usassi at e bt a 8 bit avrei overflow
-- all'atto del calcolo di 2*A, usando at e bt a 9 bit evito
-- l'overflow

architecture behav of eval is
begin
process(a,b)
variable at,bt,tmp: std_logic_vector(8 downto 0);
begin
  at:=a(7 downto 0) & '0';
  bt:=b(7) & b(7) & b(7 downto 1);
  tmp:=signed(at)+signed(bt);
  o<=std_logic_vector(tmp(7 downto 0));
end process;
end architecture;
```

Esercizio 3

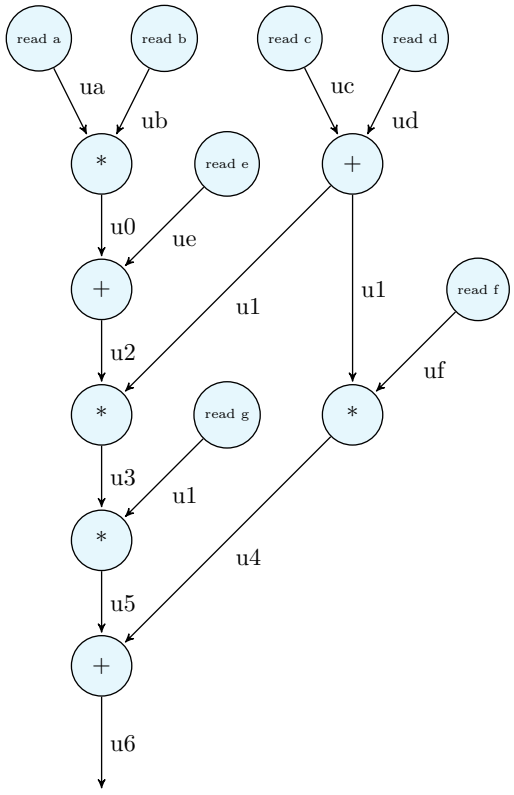
Si consideri il seguente algoritmo:

```
ua:=read a;  u2:=u0+ue;
ub:=read b;  u3:=u2*u1;
u0:=ua*ub;   uf:=read f;
uc:=read c;  u4:=u1*uf;
ud:=read d;  ug:=read g;
u1:=uc+ud;   u5:=u3*ug;
ue:=read e;  u6:=u5+u4;
```

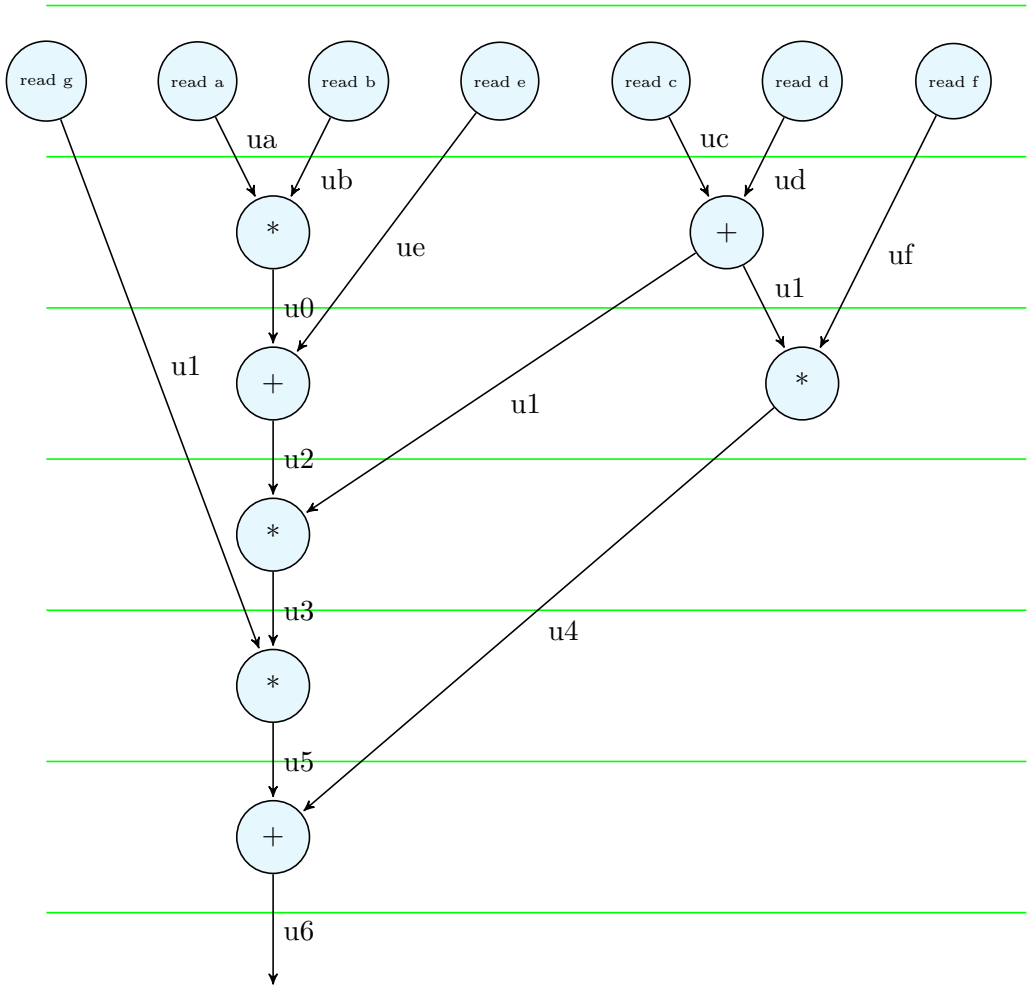
si tracci il DFG e si determinino lo scheduling ASAP e quello ALAP nell'ipotesi di ciclo singolo per tutte le operazioni, comprese quelle di lettura. Si determini poi uno scheduling, che per una latenza pari a quella minima, ottimizzi le risorse (adder,multiplier e bus). Si minimizzi poi il numero di registri utilizzati per tale scheduling (pt. 5.0).

Soluzione

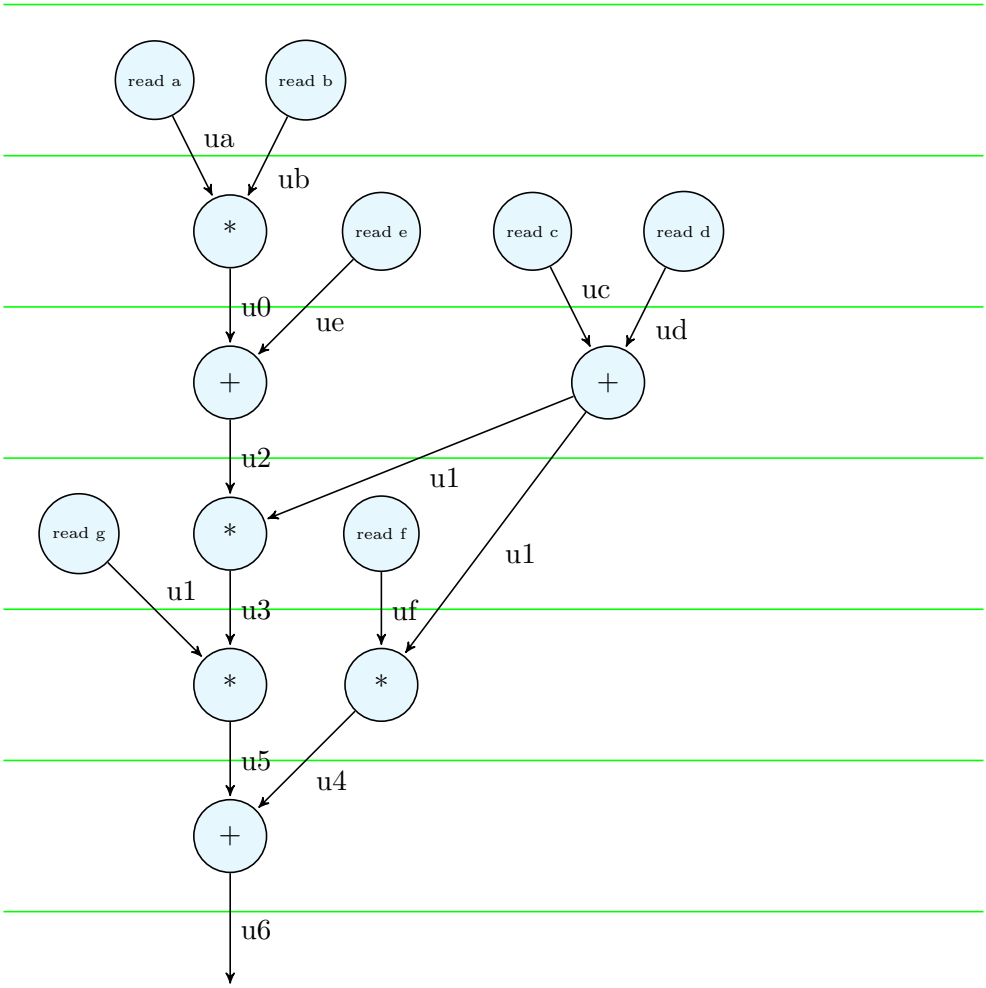
DFG dell'algoritmo



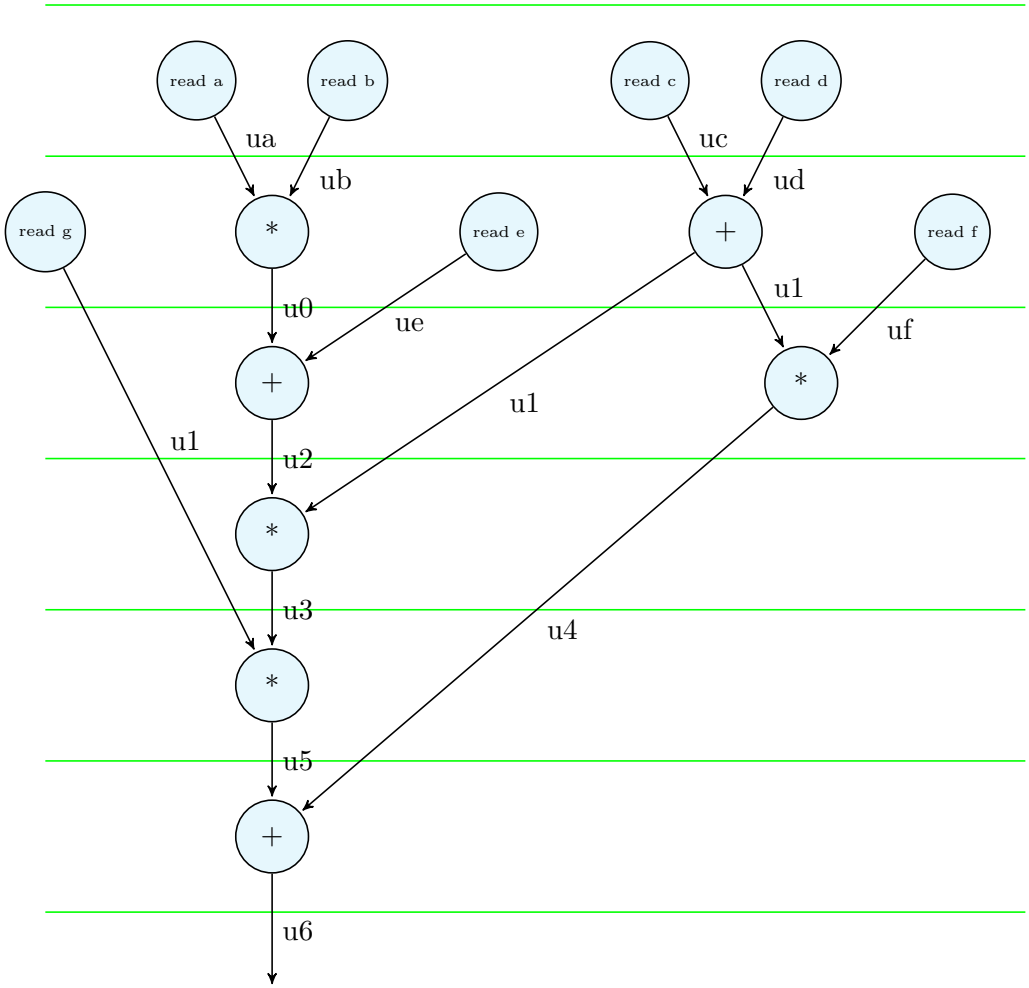
Scheduling ASAP. Allocazione: 1 sommatore, 1 moltiplicatore e 7 bus.



Scheduling ALAP. Allocazione: 2 sommatori, 2 moltiplicatori, 3 bus.



Scheduling a risorse minime che richiede l'allocazione di 1 adder, 1 moltiplicatore e 4 bus.



Grafo di incompatibilitá

Grafo colorato: 5 registri.