Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi un modello comportamentale in VHDL di un componente che riceve in ingresso un segnale x e un segnale periodico CLK. A ogni fronte positivo del clock, la rete produce in uscita y un impulso di durata pari a un periodo di clock tutte le volte che il valore campionato di x (χ) ha una transizione da '0' a '1'. In particolare, se $\chi^{k-1}\chi^k = 01$, allora $y^k = 1$ ove k é il k-mo periodo di clock.

Esempio

χ `0000111100100....`

y `0000100000100....`

Soluzione

```
library ieee;
use ieee.std_logic_1164.all;
entity pulse_gen is
  port(clk,x: in std_logic;
       y: out std_logic);
end entity pulse_gen;

architecture behav of pulse_gen is
signal x0,x1: std_logic;
begin

  process(clk)
     begin
       if (clk='1') and (clk'last_value='0') then
         x0<=x;
         x1<=x0;
       end if;
  end process;

  y<=x0 and not(x1);

end architecture;
```

Esercizio 2

Si realizzi in VHDL un modello comportamentale di un componente combinatorio con 3 parole in ingresso a, b, c ciascuna delle quali contiene 4 bit che rappresentano interi senza segno. La rete ha una parola y in uscita anch'essa di 4 bit che assume

un valore uguale al massimo fra a, b e c. I dati in ingresso e in uscita devono essere codificati con il tipo di dato `std_logic_vector`.

Soluzione

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

entity srt is
  generic(n: natural:=4);
  port (a,b,c: in std_logic_vector(n-1 downto 0);
        dat: out std_logic_vector(n-1 downto 0));
end entity srt;

architecture behav of srt is
begin

p0: process(a,b,c)
    variable xa,xb,xc: unsigned(n-1 downto 0);
    begin

      xa:=unsigned(a);
      xb:=unsigned(b);
      xc:=unsigned(c);

      if ((xa>xb) and (xa>xc)) then
        dat<=a;
      elsif ((xb>xa) and (xb>xc)) then
           dat<=b;
         else
           dat<=c;
      end if;

    end process p0;

end architecture behav;
```

Esercizio 3

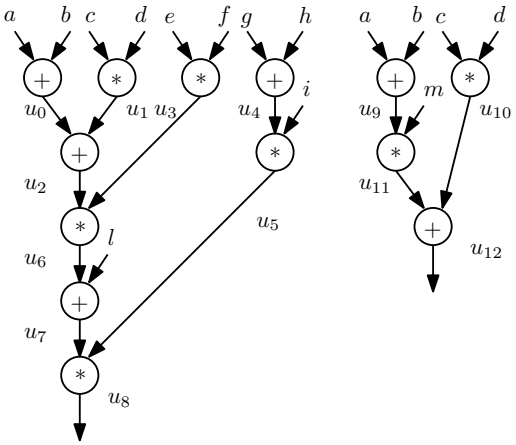
Si consideri il seguente algoritmo:

```
0. u0:=a*b;
1. u1:=c+d;
2. u2:=u0+u1;
3. u3:=e*f;
4. u4:=g+h;
5. u5:=u4*i;
6. u6:=u2*u3;
7. u7:=l+u6;
8. u8:=u5*u7;
9. u9:=a+b;
10. u10:=c*d;
11. u11:=u9*m;
12. u12:=u11+u10;
```

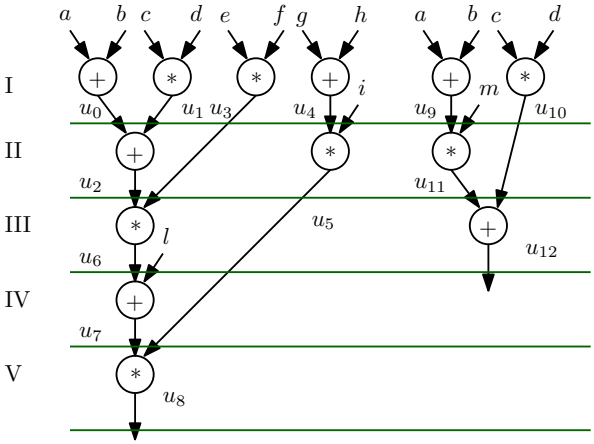
si tracci il DFG e si determinino lo scheduling ASAP e ALAP nell'ipotesi di ciclo singolo, determinando il minimo numero di risorse per ciascun tipo di operatore.

Si determini poi uno scheduling a latenza minima che ottimizza il numero di risorse. Per tale scheduling, si traccino poi il grafo di incompatibilitá delle variabili e si determini il minimo numero di registri. Si determini un binding delle risorse e si riporti la descrizione al livello RTL del componente.

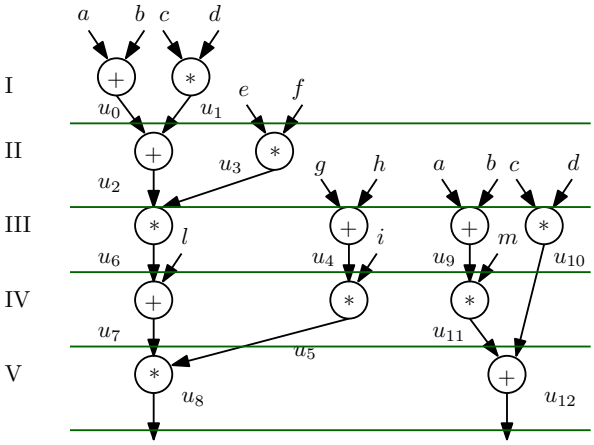
Soluzione



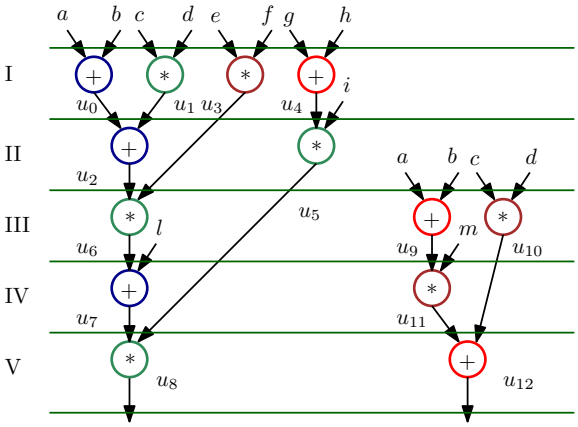
DFG



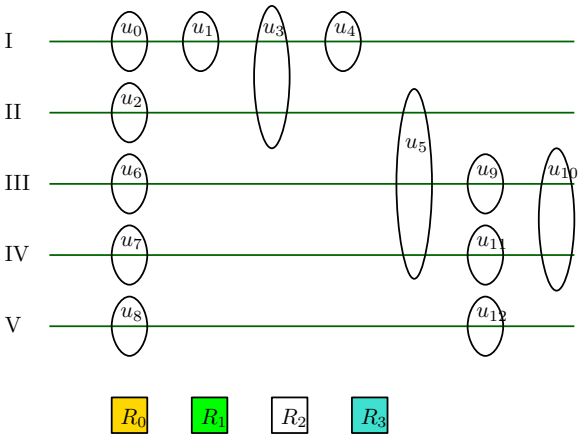
DFG con scheduling ASAP (latenza=5)



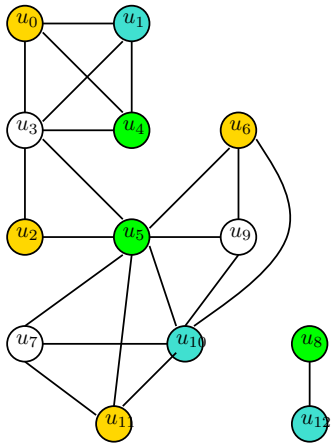
DFG con scheduling ALAP



DFG con latenza minima e risorse ottimizzate (2 multiplier e 2 adder). Il binding é indicato dai colori dei nodi del DFG.



Grafo di incompatibilitá



La descrizione RTL annotata dal binding é fornita nella seguente tabella:

cycle	mult. 1 (turchese)	mult. 2 (marron)	add. 1 (blu)	add. 2 (rosso)
I	`R3:=c*d;`	`R2:=e*f`	`R0:=a+b;`	`R1:=g+h;`
II	`R0:=R0*R3;`		`R1:=R1*i;`	
III	`R0:=R0*R2;`	`R3:=c*d;`		`R2:=a+b;`
IV		`R0:=m*R2;`	`R2:=l+R0;`	
V	`R1:=R1*R2;`			`R3:=R0*R3;`