Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi un modello comportamentale in VHDL di un flip-flop di tipo D in cui il segnale in ingresso (d) viene campionato sul fronte di salita del clock ($clock$), ma l'uscita q cambia solo sul fronte di discesa del clock. Il modello deve tenere conto del tempo di setup (τ_{dc}) rispetto al fronte di salita del clock e del tempo di risposta ($\tau_{c'q}$) rispetto al fronte di discesa del clock.

Soluzione

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

entity dff_hp is
    generic(tdc,tncq: time);
    port(d,clk: in std_logic;
         q: out std_logic);
end entity dff_hp;

architecture behav of dff_hp is
signal tmp: std_logic;
begin
  process(clk)
    begin
      if ((clk='1') and (clk'last_value='0')) then
        if (d'stable(tdc)) then
          tmp <= d;
        else
          tmp <= 'X';
        end if;
      elsif ((clk='0') and (clk'last_value='1')) then
        q <= tmp after tncq;
      end if;
  end process;
end architecture behav;
```

Esercizio 2

Si realizzi un modello comportamentale in VHDL di una rete combinatoria che riceve in ingresso una parola $x_{7..0}$ che rappresenta un numero intero positivo (k) e produce in uscita una parola $y_{7..0}$ che rappresenta il complemento a 2 di k.

Soluzione

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

entity cmpl2 is
  port(x: in std_logic_vector(7 downto 0);
       y: out std_logic_vector(7 downto 0));
end entity cmpl2;

architecture behav of cmpl2 is
begin
  process(x)
  variable w: unsigned(7 downto 0);
  begin
    w:=unsigned(not(x));
    y <= std_logic_vector(w + "00000001");
  end process;
end architecture behav;
```

Esercizio 3

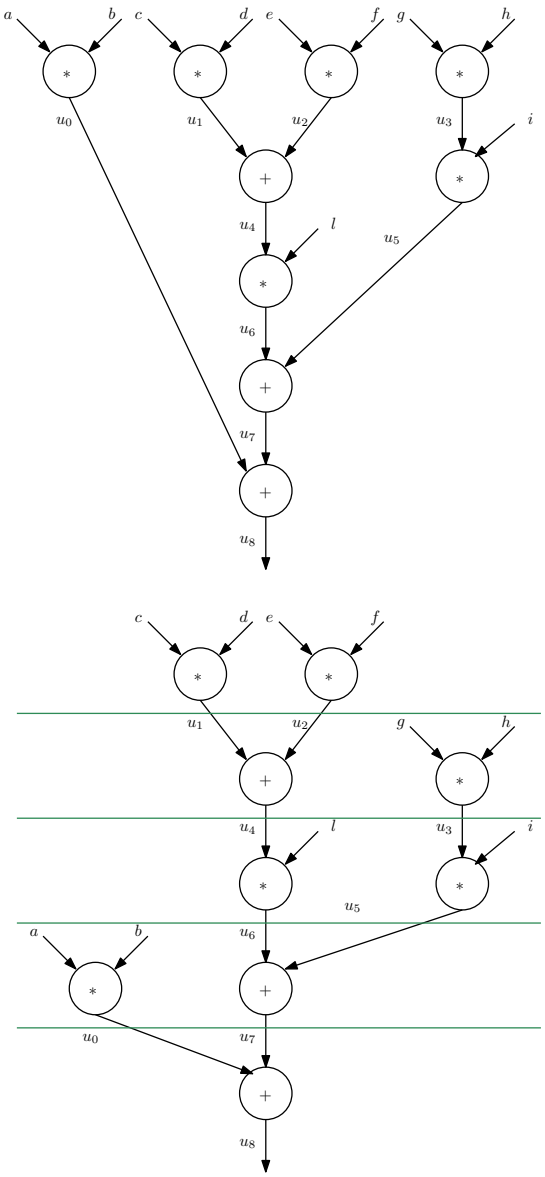
Si consideri il seguente algoritmo:

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

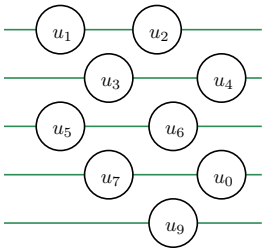
si tracci il DFG e si determini uno scheduling a latenza minima, nell'ipotesi di ciclo singolo, che minimizzi il numero di risorse utilizzate. Si indichino allocazione e binding. Si determini poi il minimo numero di registri utilizzabile e si descrivano le operazioni svolte al livello RTL. Supponendo poi che il ritardo massimo di un moltiplicatore sia di 6ns e quello di un sommatore sia di 1.4ns, si determini, trascurando i ritardi di multiplexer e flip-flop, la latenza e la banda (throughput) di tale data-path.

Soluzione

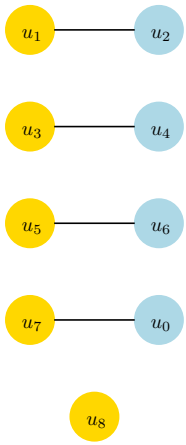
DFG e DFG con scheduling ottimizzato (allocazione: 1 adder e 2 moltiplicatori, latenza relativa: 5)



Ottimizzazione del numero di registri.



tempo di vita delle variabili



grafo di incompatibilitá

Descrizione RTL semplificata (R0 giallo, R1 azzurro)

clock	mult0	mult1	add
I	$R_0 = c * d$	$R_1 = e * f$	
II	$R_0 = g * h$		$R_1 = R - 0 + R_1$
III	$R_0 = R_0 * i$	$R_1 = R_1 * l$	
IV		$R_1 = a * b$	$R_0 = R_0 + R_1$
V			$R_0 = R_0 + R_1$

Segue una descrizione schematica del data-path (che nel compito non era richiesta)