**Compito di linguaggi di descrizione dell'hardware**

**Esercizio 1**

Si realizzi un modello comportamentale in VHDL di un lath di tipo D trasparente (con ingressi $d$ e $clk$, ed uscita $q$). Quando $clk = 1$, il latch é nella fase di SAMPLE e quando $clk = 0$, il latch é nella fase di HOLD. Il modello deve descrivere il tempo di risposta del componente ($tr$). Inoltre, é necessario descrivere il possibile degrado dell'informazione memorizzata che si ha se il componente rimane per un tempo maggiore di $tx$ nello stato di HOLD (é un problema dei latch dinamici). In tale caso, l'uscita $q$ si porta a $X$.

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

-- model of a transparent (dynamic) dlatch with delay (tdq) tr and
-- tx as the maximum data retention time in the hold state

entity dlatch is
  generic(tr,tx: time);
  port(d,clk: in std_logic;
       q: out std_logic);
end entity dlatch;

architecture behav of dlatch is
begin
  p0: process(d,clk)
      begin
        if (clk='1') then
          q<=d after tr;
        elsif (clk'event) and (clk='0') then
            q<='X' after tx;
        end if;
      end process p0;
end architecture behav;
```

**Esercizio 2**

Si descriva al livello comportamentale in VHDL un componente che esegue operazioni di rotate. Questo riceve in ingresso una parola da 8 bit ($w_{7..0}$), una parola di 3 bit ($ra_{2..0}$) che codifica un intero positivo $RA$ e 2 bit $L$ e $R$ che codificano la direzione del rotate. Se $LR = 10$ il dato in ingresso viene rotato verso sinistra di $RA$ posizioni,

se $LR = 01$ il dato in ingresso viene rotato verso destra di $RA$ posizioni, se $LR = 00$ il dato passa inalterato.

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

entity rotate is
   generic(delay: time);
   port(w: in std_logic_vector(7 downto 0);
        ra: in std_logic_vector(2 downto 0);
        l,r: in std_logic;
        y: out std_logic_vector(7 downto 0));
end entity rotate;

architecture behav of rotate is

begin

process(w,ra,l,r)
variable ymp: std_logic_vector(7 downto 0);
begin

ymp:=w;
if (l='0') and (r='1') then
  if (ra(0)='1') then
    ymp := ymp(0) & ymp(7 downto 1);
  end if;
  if (ra(1)='1') then
    ymp := ymp(1 downto 0) & ymp (7 downto 2);
  end if;
  if (ra(2)='1') then
    ymp := ymp(3 downto 0) & ymp (7 downto 4);
  end if;
elsif (l='1') and (r='0') then
     if (ra(0)='1') then
       ymp := ymp(6 downto 0) & ymp(7);
     end if;
     if (ra(1)='1') then
       ymp := ymp (5 downto 0) & ymp(7 downto 6);
     end if;
     if (ra(2)='1') then
       ymp := ymp (3 downto 0) & ymp(7 downto 4);
```

```
     end if;
     elsif not((l='0') or (r='0')) then
          ymp:="XXXXXXXX";
end if;
y <= ymp;
end process;

end architecture behav;
```

**Esercizio 3**

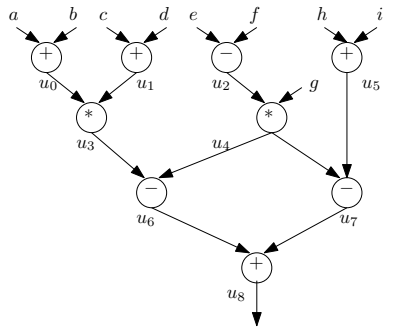
Si consideri il seguente algoritmo:

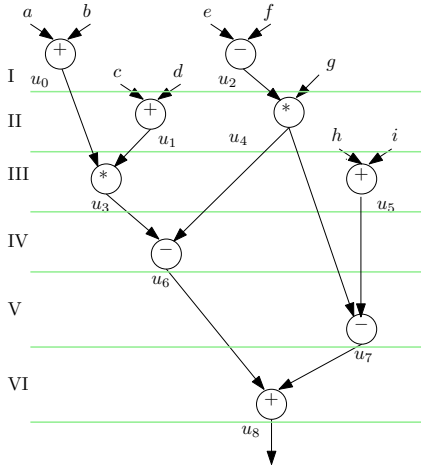
```
0. u0:=a+b;
1. u1:=c+d;
2. u2:=e-f;
3. u3:=u0*u1;
4. u4:=u2*g;
5. u5:=h+i;
6. u6:=u3-u4;
7. u7:=u4-u5;
8. u8:=u7+u6;
```

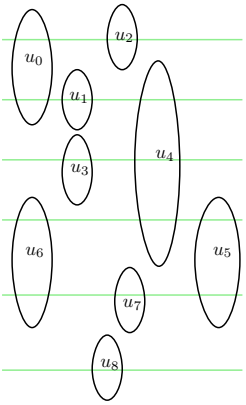
Si determini uno scheduling che, utilizzando come risorse un sommatore, un sottrattore e un moltiplicatore, minimizzi la latenza. Per tale scheduling si determini il minimo numero di registri da utilizzare e si ottenga una descrizione al livello RTL.

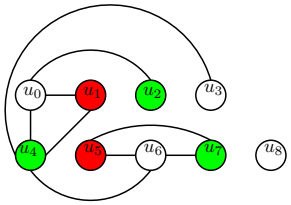
Si supponga poi che invece di sommatori e sottrattori siano disponibili due ALU in grado di eseguire somma o sottrazione. Si determini anche in questo caso uno scheduling che utilizzando un moltiplicatore e le due ALU minimizzi la latenza. In tutti i casi si faccia l'ipotesi di ciclo singolo.

**Soluzione**



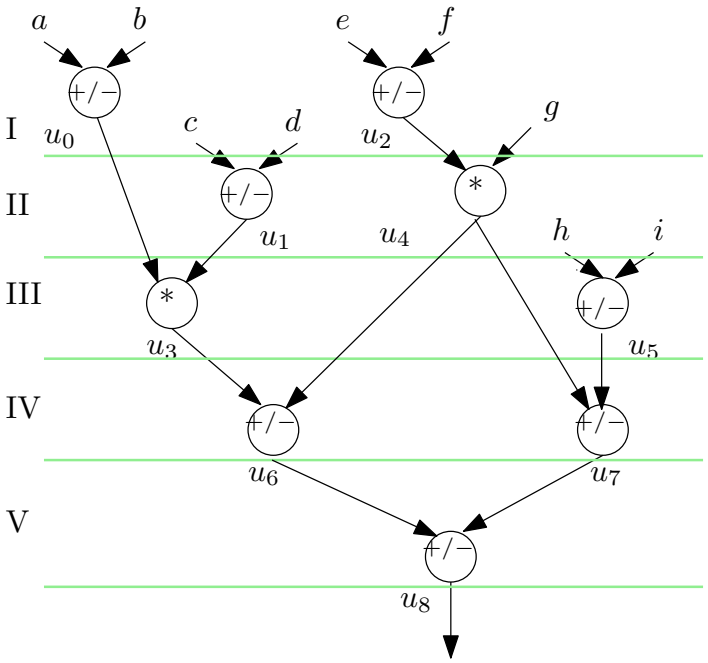




$R_0 = \{u_0, u_3, u_6, u_8\}$
$R_1 = \{u_1, u_5\}$
$R_2 = \{u_2.u_4, u_7\}$

| RTL description clock period | adder | subtractor | multiplier |
|---|---|---|---|
| I | $R_0 := a + b$ | $R_2 := e - f$ | |
| II | $R_1 := c + d$ | | $R_2 := R_2 * g$ |
| III | $R_1 := h + i$ | | $R_3 := R_0 * R_1$ |
| IV | | $R_0 := R_0 - R_2$ | |
| V | | $R_2 := R_2 - R_1$ | |
| VI | $R_0 := R_0 + R_2$ | | |

Utilizzando le 2 ALU si nota dal precedente DFG con scheduling che la sottrazione al ciclo 6 puó essere anticipata al ciclo 5. E quindi la latenza diviene ora di 5 cicli di clock.


a
b
e
f
+/−
+/−
I
u0
c
d
u2
g
+/−
*
II
u1
u4
h
i
III
*
+/−
u3
u5
IV
+/−
+/−
u6
u7
V
+/−
u8