Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi il modello comportamentale in VHDL di un flip-flop di tipo JK che campiona sul fronte di salita. L'equazione caratteristica di tale dispositivo é: $Q_{i+1} = J_i Q_i' + K_i' Q_i$. Il modello deve: i) verificare eventuali violazioni del tempo di setup (t_{SU}) da parte degli ingressi; ii) produrre il dato campionato in uscita con un ritardo t_R. Entrambi i parametri devono essere passati come generic.

Soluzione

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

entity ffjk is
    generic(tsu,tr: time);
    port(clk: in std_logic;
         j,k: in std_logic;
         q: out std_logic);
end entity;

architecture behav of ffjk is
begin
  process(clk)
  variable state: std_logic;
  begin
    if (rising_edge(clk)) then
      if (j'stable(tsu) and k'stable(tsu)) then
        state:=(j and not(state)) or (not(k) and state);
      else
        state:='X';
      end if;
      q<=state after tr;
    end if;
  end process;
end architecture;
```

Esercizio 2

Si realizzi un modello comportamentale in VHDL di una rete combinatoria che riceve in ingresso tre parole di n bit ciascuna $a_{n-1..0}$, $b_{n-1..0}$ e $c_{n-1..0}$ che rappresentano numeri interi positivi. Compito della rete é calcolare $y = a + 2b + 3c$ dove y é rappresentato su $n+2$ bit. Non si puó utilizzare l'operatore prodotto (*), se possibile si minimizzi il numero di somme.

Soluzione

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

entity adder is
    generic(n: natural);
    port(a,b,c: in std_logic_vector(n-1 downto 0);
         y: out std_logic_vector(n+2 downto 0));
end entity;

architecture behav of adder is
begin
  process(a,b,c)
  variable vara,var2b,var2c,varc: unsigned(n+2 downto 0);
  begin
    vara:=unsigned("000" & a);
    var2b:=unsigned("00" & b & '0');
    var2c:=unsigned("00" & c & '0');
    varc:=unsigned("000" & c);
    y <= std_logic_vector(vara+var2b+var2c+varc);
  end process;
end architecture;
```

Esercizio 3

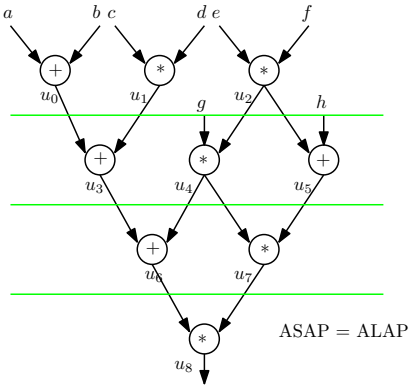
Si consideri il seguente algoritmo:

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

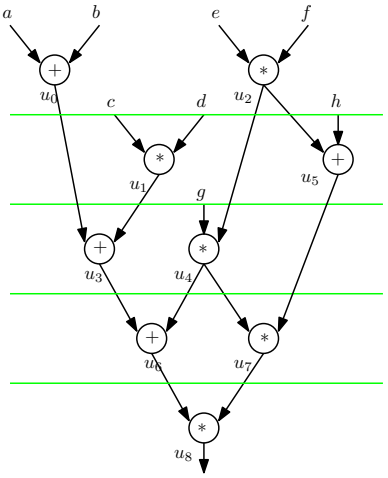
si tracci il DFG e si determinino lo scheduling ASAP e quello ALAP indicando latenza e risorse allocate per ciascuno scheduling. Si determini poi uno scheduling a risorse assegnate (un moltiplicatore e due adder) che minimizzi la latenza. Si determini poi il minimo numero di registri da utilizzare e si descrivano le operazioni svolte al livello RTL.

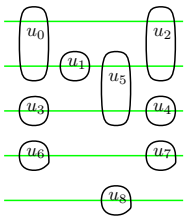
Soluzione

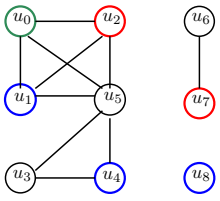
u0:=a+b; u1:=c*d; u2:=e*f; u3:=u0+u1; u4:=u2*g;
u5:=u2+h; u6:=u3+u4; u7:=u4*u5; u8:=u7*u6;



ASAP = ALAP







Servono 4 registri: bianco $R_0 = \{u_3, u_5, u_6\}$, rosso $R_1 = \{u_2, u_7\}$, blu $R_2 = \{u_1, u_4, u_8\}$, verde $R_3 = \{u_0\}$.

Descrizione RTL

cycle	add	mult
1	$R_3 = a + b$	$R_1 = e * f$
2	$R_0 = R_1 + h$	$R_2 = c + d$
3	$R_0 = R_3 + R_2$	$R_2 = R_1 * g$
4	$R_0 = R_0 + R_2$	$R_1 = R_2 * R_0$
5		$R_2 = R_0 * R_1$