Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi un modello comportamentale in VHDL di un componente sincrono che, oltre al segnale di clock, ha un ingresso D e un uscita Q. Il componente campiona il valore di D ogni due fronti di salita del segnale di clock. Il tempo di risposta (t_{CQ}) viene passato come generic.
Soluzione

```
entity prova is
  generic(tr: time);
  port(d: in std_logic;
       clk: in std_logic;
       q: out std_logic);
end entity prova;

architecture behav of prova is
signal t2: std_logic:='0'; -- auxiliary clock
begin

  process(clk)
  begin
    if (rising_edge(clk)) then
      t2<=not t2;
    end if;
  end process;

  process(t2)
  begin
    if (rising_edge(t2)) then
      q <= d after tr;
    end if;
  end process;

end architecture behav;
```

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 senza segno A e B. Compito della rete é produrre in uscita $o_{7..0}$ il valore di $A + 7 * B$ se $A + 7 * B$ é rappresentabile con 8 bit e $2^8 - 1$ se il risultato della somma non é rappresentabile con 8 bit.

Soluzione

Il massimo valore di $A + 7 * B$ é $255 + 255 * 7 = 2040$ che é rappresentabile con 11 bit.

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

entity module is
  port(a,b: in std_logic_vector(7 downto 0);
       o: out std_logic_vector(7 downto 0));
end entity module;

architecture behav of module is
begin
  process(a,b)
  variable otmp: unsigned(10 downto 0);
  begin
   otmp:=unsigned("000"&a)+unsigned("000"&b);   -- a+b
   otmp:=otmp+unsigned("00"&b&'0');   -- a+b+2b
   otmp:=otmp+unsigned("0"&b&"00");   -- a+b+2b+4b

   if (otmp(10 downto 8)/="000") then
     otmp:="00011111111";
   end if;

   o<=std_logic_vector(otmp(7 downto 0));
  end process;
end architecture;
```

Esercizio 3

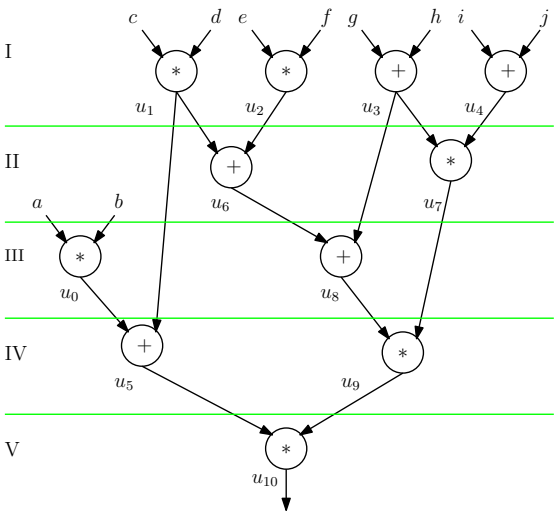
Si consideri il seguente algoritmo:

```
u0:=a*b;
u1:=c*d;
u2:=e*f;
u3:=g+h;
u4:=i+j;
u5:=u1+u0;
u6:=u2+u1;
u7:=u3*u4;
u8:=u6+u3;
u9:=u8*u7;
u10:=u5*u9;
```

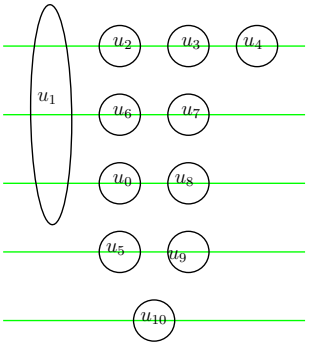
si tracci il DFG e si determini poi uno scheduling che, con l'ipotesi di ciclo singolo, minizimizzi il numero di risorse utilizzate. Si determini poi il minimo numero di registri da utilizzare. Si discuta sinteticamente l'effetto della minimizzazione del numero dei registri sul costo della rete e sulla massima frequenza di clock utilizzabile.

Soluzione

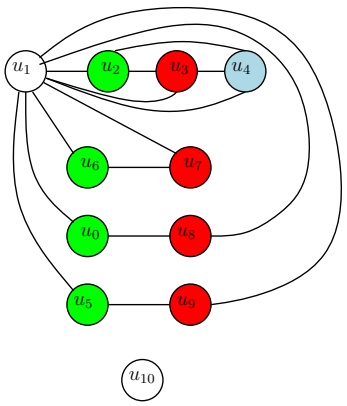
DFG scheduled a latenza minima e con risorse ottimizzate (due moltiplicatori e due adder)



tempo di vita delle variabili



grafo di incompatibilitá



La minimizzazione del numero dei registri riduce il numero di registri rispetto all'algoritmo iniziale, questa minimizzazione si puó ridurre parzialmente in una riduzione di costo a causa dell'utilizzo di multiplexer che servono per condividere tali risorse. L'effetto sul ritardo é sempre dovuto ai multiplexer aumentano la profonditá logica dei cammini critici e quindi possono aumentare il ritardo.