Compito di linguaggi di descrizione dell'hardware

Esercizio 1

Si realizzi un modello comportamentale in VHDL di un n-bit adder che lavora in aritmetica saturata. Il componente ha 2 parole in ingresso a e b di n-bit ciascuna che rappresentano interi senza segno e un uscita sum di n bit. L'uscita é uguale ad $a + b$ se $a + b \leq 2^n - 1$ e a $2^n - 1$ se $a + b > 2^n - 1$. In caso di difficoltá, l'esercizio puó essere semplificato fissando un valore per n.

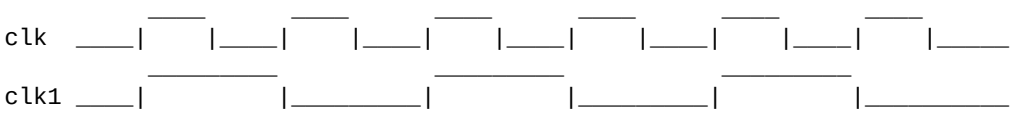
```
library ieee;
use ieee.std_logic_1164.all;
constant n: integer;

entity saturated_adder is
      port(a,b: in std_logic_vector(n-1 downto 0);
           sum: out std_logic_vector(n-1 downto 0));
end entity saturated_adder;

architecture behav of saturated adder is
begin
  process(a,b)
  variable a1,b1,sum1: unsigned(n downto 0);
  begin
    a1:=unsigned('0' & a);
    b1:=unsigned('0' & b);
    sum1:=a1+b1;
    if (sum1(n)='0') then
      sum<=std_logic_vector(sum1(n-1 downto 0));
    elsif (sum1(n)='1') then
         sum<=(others <= '1');
    end if;
  end process;
end architecture behav;
```

Esercizio 2

Si descriva al livello comportamentale in VHDL una rete che ha un ingresso clk e che produce in uscita due forme d'onda $clk1$ e $clk2$. Supponendo che clk sia periodico, $clk1$ ha un periodo pari a due volte quello di clk e $clk2$ pari a 4 volte.




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

entity div is
  port(clk: in std_logic;
       clk1,clk2: out std_logic);
end entity div;

architecture behav of div is
signal tmp1, tmp2: std_logic:='0';
begin

p0: process(clk)
    begin
      if (rising_edge(clk)) then
        tmp1<=not(tmp1);
      end if;
    end process p0;

p1: process(tmp1)
    begin
      if (rising_edge(tmp1)) then
        tmp2<=not(tmp2);
      end if;
    end process p1;

    clk1<=tmp1;
    clk2<=tmp2;

end architecture behav;
```

Esercizio 3

Si consideri il seguente algoritmo:

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

Si tracci il DFG e si determinino lo scheduling ASAP e quello ALAP. Si determini poi uno scheduling che utilizza il minimo numero di risorse (1 adder e 1 moltiplicatore) e che ottimizza la latenza (nell'ipotesi di ciclo singolo). Si determini poi il numero minimo di registri da utilizzare, il binding delle risorse e si descriva il componente al livello RTL.