Esercizio 1
Sia $a \leftrightarrow A$ su 8 bit in base 2. Determinare il numero minimo di bit n necessari per rappresentare il numero $-6a$. Indicando con B la rappresentazione di $-6a$ su n bit, progettare la rete che riceve in ingresso A e produce in uscita *B, senza fare uso di moltiplicatori*.

Soluzione
Il campo di rappresentazione di a è [-128,127], quindi:

$-6a \in [-762,768] \subset [-2^{10}, 2^{10}-1]$,

da cui si deduce $n = 11$.

La rete si può progettare osservando che $-6a = 2(a - 4a)$, quindi:

$$B = \left|2 \cdot (a - 4a)\right|_{2^{11}} = 2 \cdot \left|\left|a\right|_{2^{10}} - \left|4a\right|_{2^{10}}\right|_{2^{10}} = 2 \cdot \left|A^{EST} - \left|4A\right|_{2^{10}}\right|_{2^{10}}$$

dove $A^{EST} \leftrightarrow a$ su 10 bit in base 2 (il secondo passaggio è giustificato dal fatto che la rappresentazione di $a - 4a$ è riducibile su 10 bit).

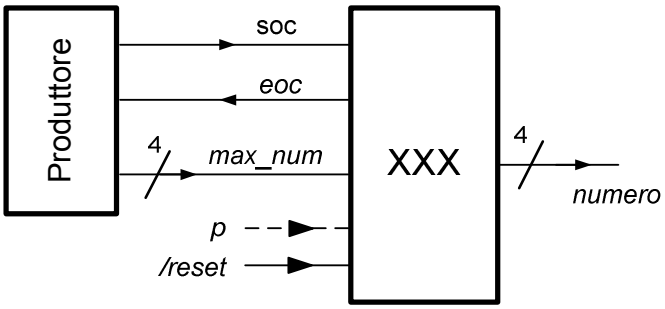
Una possibile descrizione della rete è la seguente:

```
module RCesercizio1(B,A);
input   [7:0] A;
output [10:0] B;
wire    [9:0] w;
assign w = {A[7],A[7],A}-{A,2'B00};
assign B = {w,1'B0};
endmodule
```

Esercizio 2
Descrivere e sintetizzare l'unità **XXX** in modo che risponda alle seguenti specifiche:

1) tramite l'uscita *numero* emette un numero naturale che, partendo da 0 cresce di uno ad ogni ciclo di clock;
2) contemporaneamente si fa inviare dal Produttore un numero naturale *max_num* che interpreta come il massimo numero che va emesso tramite *numero*
3) quando il massimo numero è stato emesso, torna al punto1.

Si supponga che mentre XXX sta emettendo i numeri, ci sia tutto il tempo per un handshake completo con il Produttore e quindi per ricevere e utilizzare subito il nuovo numero naturale *max_num*.



Fare un diagramma temporale che illustri un ciclo completo di evoluzione di *XXX*, supponendo che la risposta Produttore, una volta iniziato l'handshake, avvenga tra uno e due cicli del clock di *XXX* e che il Produttore fornisca *max_num*=7;

Soluzione

```
module XXX(soc,eoc,max_num,numero,p,reset_);
 input       p,reset_;
 input       eoc;
 output      soc;
 input  [3:0] max_num;
 output [3:0] numero;
 reg          SOC;               assign soc=SOC;
 reg [3:0]    MAX_NUM, NUMERO;   assign numero=NUMERO;
 reg [1:0]    STAR;    parameter S0=0,S1=1,S2=2,S3=3;
 always @(posedge p or negedge reset_) if (reset_==0) begin SOC=0; STAR=S0; end
  else #3
   casex(STAR)
    S0:  begin NUMERO<=0; SOC<=1;STAR<=S1; end
    S1:  begin NUMERO<=NUMERO+1; STAR<=(eoc==0)?S2:S1; end
    S2:  begin NUMERO<=NUMERO+1; SOC<=0; MAX_NUM<=max_num;
               STAR<=(eoc==0)?S2:S3; end
    S3:  begin NUMERO<=NUMERO+1; STAR<=(NUMERO==MAX_NUM-1)?S0:S3; end
   endcase
endmodule
```