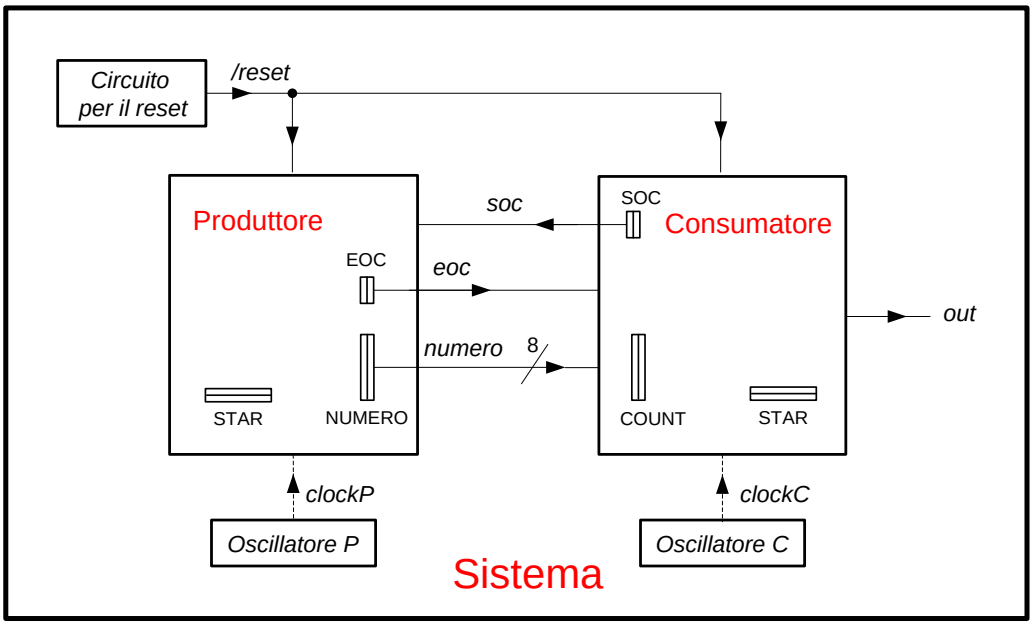
Sistema con Produttore e Consumatore

handshake *soc_* (*start of computation*), *eoc* (*end of computation*)




```
module Sistema;
 // Terminazione forzata dopo 225 ns
 initial begin #225 $stop; end
 // Generatore del reset
 reg reset_; initial begin reset_=0; #1 reset_=1; end
 // Oscillatore per il clock del Produttore con periodo di 12 ns
 reg clockP ; initial clockP=0; always #6 clockP<=(!clockP);
 // Oscillatore per il clock del Consumatore con periodo di 8 ns
 reg clockC ; initial clockC=0; always #4 clockC<=(!clockC);
 // Collegamenti tra moduli
 wire[7:0] numero;
 wire soc,eoc;
 // Variabile di uscita del Consumatore;
 wire out;
 // Moduli
 Produttore     PRO(soc,eoc,numero,clockP,reset_);
 Consumatore    CON(soc,eoc,numero,out,clockC,reset_);
endmodule
```

Lo ***schema di evoluzione del Produttore*** che useremo è molto semplice (ma non è il più efficiente)

1. Segnalazione di fine della computazione precedente e attesa della richiesta, da parte del Consumatore, di una nuova computazione.
2. Segnalazione di inizio della computazione e attesa della rimozione, da parte del Consumatore, della richiesta della computazione.
3. Computazione e Ritorno al punto 1.

Lo ***schema di evoluzione del Consumatore*** che useremo è molto semplice (ma non è il più efficiente)

1. Richiesta di nuova computazione e attesa che il Produttore segnali che ha capito e che sta per iniziare la computazione richiesta.
2. Rimozione della richiesta di computazione, attesa che il Produttore segnali di aver terminato la computazione e prelievo del risultato della computazione stessa.
3. Utilizzo del risultato della computazione e ritorno al punto 1.

```
// Produttore di numeri dispari ad 8 bit
module Produttore(soc,eoc,numero,clockP,reset_);
 input        clockP,reset_;
 input        soc;
 output       eoc;
 output [7:0] numero;
 reg          EOC;      assign eoc=EOC;
 reg [7:0]    NUMERO;   assign numero=NUMERO;
 reg [1:0]    STAR;     parameter S0=0,S1=1,S2=2;

 always @(reset_==0) begin EOC<=1; NUMERO<=255; STAR<=S0; end
 always @(posedge clockP) if (reset_==1) #3
   casex(STAR)
    S0:  begin EOC<=1; STAR<=(soc==1)?S1:S0; end
    S1:  begin EOC<=0; STAR<=(soc==1)?S1:S2; end
    S2:  begin NUMERO<=NUMERO+2; STAR<=S0; end
   endcase
endmodule


// Formatore di impulsi di durata, in periodi di clock, pari al numero ricevuto
module Consumatore(soc,eoc,numero,out,clockC,reset_);
 input        clockC,reset_;
 input        eoc;
 output       soc;
 input[7:0]   numero;
 output       out;
 reg          SOC;  assign soc=SOC;
 reg          OUT;  assign out=OUT;
 reg[7:0]     COUNT;
 reg[1:0]     STAR; parameter S0=0,S1=1,S2=2;
 always @(reset_==0) #1 begin SOC<=0; OUT<=0; STAR<=S0; end
 always @(posedge clockC) if (reset_==1) #3
   casex(STAR)
    S0: begin SOC<=1; OUT<=0; STAR<=(eoc==1)?S0:S1; end
    S1: begin SOC<=0; COUNT<=numero; STAR<=(eoc==0)?S1:S2; end
    S2: begin OUT<=1; COUNT<=COUNT-1; STAR<=(COUNT==1)?S0:S2; end
   endcase
endmodule
```