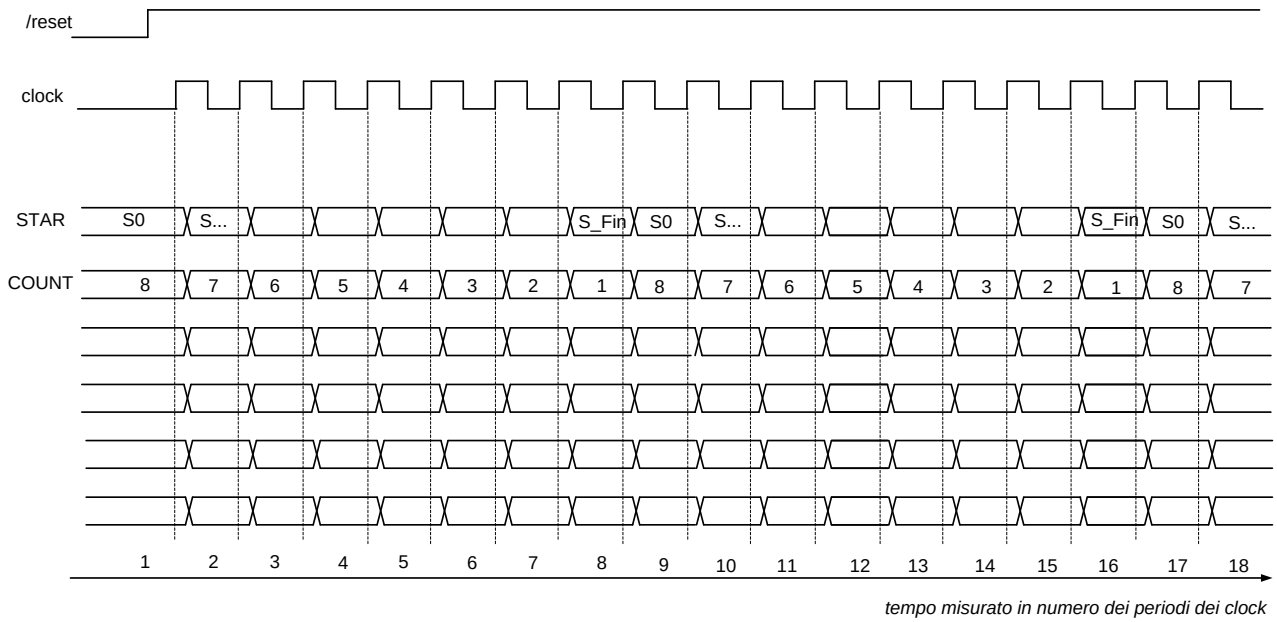
Due schemi di descrizione di un circuito che ripete ciclicamente, con un ritmo di **……** periodi di clock, la stessa elaborazione.

//Schema N. 1

```
module XXX(...., clock, reset_);
 input clock,reset_;
 .......
 reg [...:0] COUNT;
 .......
 reg [...:0] STAR;
 parameter S0=0, S1=1,....., S_Finale=....;
 parameter Num_Periodi=....; //inserire il numero dei periodi di clock
always @(reset_==0) begin COUNT<=Num_Periodi; .... ; STAR<=S0; end
 always @(posedge clock) if (reset_==1) #3
   casex (STAR)
    S0: begin COUNT<=COUNT-1; elaborazioni varie; ..... ; end
    S1: begin COUNT<=COUNT-1; elaborazioni varie; ..... ; end
    ...
    Altri stati con elaborazioni varie in cui viene ripetuto COUNT<=COUNT-1
    ...
    S_Prefinale: begin COUNT<=COUNT-1; elaborazioni varie; STAR<=S_Finale; end
    S_Finale:    begin COUNT<=(COUNT==1)?Num_Periodi:(COUNT-1);
                       STAR<=(COUNT==1)?S0:S_Finale; end
   endcase
endmodule
```

In tale schema, l'arrivo del primo fronte in salita del clock dopo il reset iniziale, va considerato come la fine del primo periodo di clock.

Nel diagramma che segue si esemplifica con *Num_Periodi*=8



ATTENZIONE

Se il numero di periodi non è una costante, ma dipende dal valore di una variabile di ingresso o dal contenuto di un registro, sostituire lo statement **`parameter`** con:

```
wire[ :0] Num_Periodi; assign Num_Periodi= φ(...);
```

dove φ(...) è l'espressione che fornisce il numero dei periodi di clock.

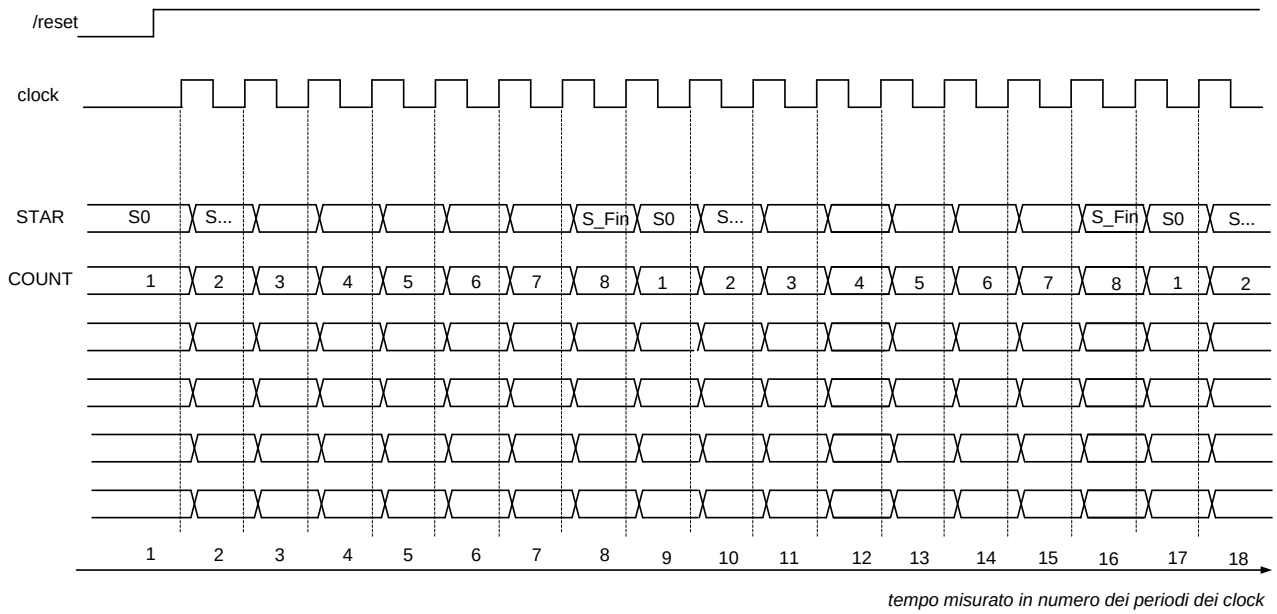
Al **reset** mettere tuttavia `COUNT<='B....` calcolando numericamente il numero dei periodi iniziale (al reset si possono inserire solo dei bit 0 o dei bit 1 agendo su `/preclear` e su `/preset`)

//Schema N. 2

```
module XXX(...., clock, reset_);
 input clock,reset_;
 .......
 reg [...:0] COUNT;
 .......
 reg [...:0] STAR;
 parameter S0=0, S1=1,....., S_Finale=....;
 parameter Num_Periodi=....; //inserire il numero dei periodi di clock
 always @(reset_==0) begin COUNT<=1; .... ; STAR<=S0; end
 always @(posedge clock) if (reset_==1) #3
   casex (STAR)
    S0      : begin COUNT<=COUNT+1; elaborazioni varie; ..... ; end
    S1      : begin COUNT<=COUNT+1; elaborazioni varie; ..... ; end
    ...
    Altri stati con elaborazioni varie in cui viene ripetuto COUNT<=COUNT+1
    ...
    S_Prefinale: begin COUNT<=COUNT+1; elaborazioni varie; STAR<=S_Finale; end
    S_Finale:    begin COUNT<=(COUNT==Num_Periodi)?1:(COUNT+1);
                       STAR<=(COUNT==Num_Periodi)?S0:S_Finale; end
   endcase
endmodule
```

In tale schema, l'arrivo del primo fronte in salita del clock dopo il reset iniziale, va considerato come la fine del primo periodo di clock.

Nel diagramma della pagina successiva si esemplifica con *Num_Periodi*=8



ATTENZIONE

Se il numero di periodi non è una costante, ma dipende dal valore di una variabile di ingresso o dal contenuto di un registro, sostituire lo statement **`parameter`** con:

```
wire[ :0] Num_Periodi; assign Num_Periodi= φ(...);
```

dove φ(...) è l'espressione che fornisce il numero dei periodi di clock.