Esercizio 1

1) Descrivere, secondo un modello campionatore-ritardatore il *flip-flop T*. Questo è un circuito con due ingressi, *t* (*toggle*) e *p*. Il FF-T si distingue dal FF-D-ET perché sul fronte in salita di *p* campiona il valore di *t,* e *conserva* se *t*=0 o *commuta* se *t*=1.

2) Sintetizzare il campionatore, utilizzando un elemento neutro come elemento di marcatura. Detto t_{CN} il tempo di attraversamento della rete combinatoria CN1, dimensionare:

a) il ritardo minimo richiesto per il meccanismo di marcatura

b) il tempo di permanenza minimo di uno stato di ingresso

3) Sintetizzare in maniera euristica un FF-T a partire da:

a) un FF-D-ET

b) un FF-JK

Esercizio 2

Descrivere e sintetizzare l'unità *XXX* di Fig. 1 nelle seguenti ipotesi:

L'unità *XXX* è normalmente a riposo e invia al convertitore D/A un byte che il convertitore traduce in $v(t)=0$. Quando l'unità *XXX* riceve dal produttore una nuova coppia (segno *s*, altezza *H*) gestisce la variabile di uscita *out* in modo da indurre il convertitore D/A a generare, tramite $v(t)$, un segnale triangolare di altezza $k \cdot H$ (con *k* costante caratteristica del Convertitore) e di polarità positiva se *s* vale 0 e negativa se *s* vale 1. Il Convertitore D/A lavora in binario bipolare (cioè interpreta *out* come la rappresentazione in traslazione di un intero) .

NOTE:

- Si faccia attenzione che *H* è un numero naturale su 7 bit
- Si convenga che *H* è sempre maggiore di 1
- Nessuna descrizione o schema debbono essere fatti relativamente al Produttore e al Convertitore.

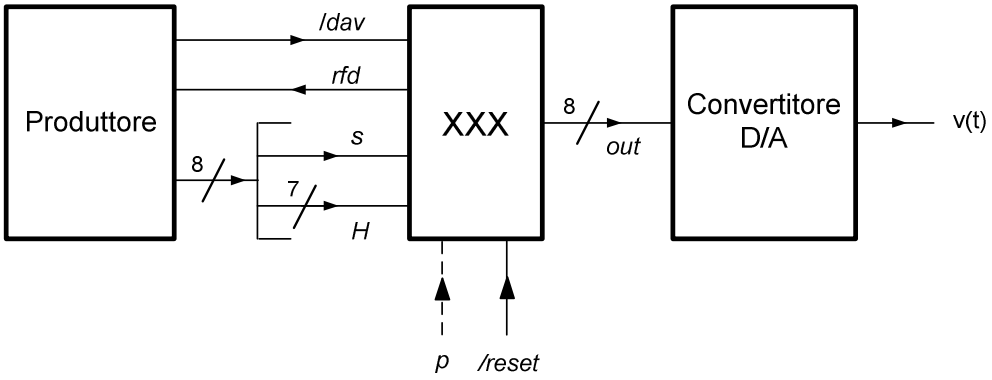


Fig. 1

In Fig. 2 è esemplificata una evoluzione consistente prima con il caso *s*=0 e *h*=2 e poi con il caso *s*=1 e *h*=2 .

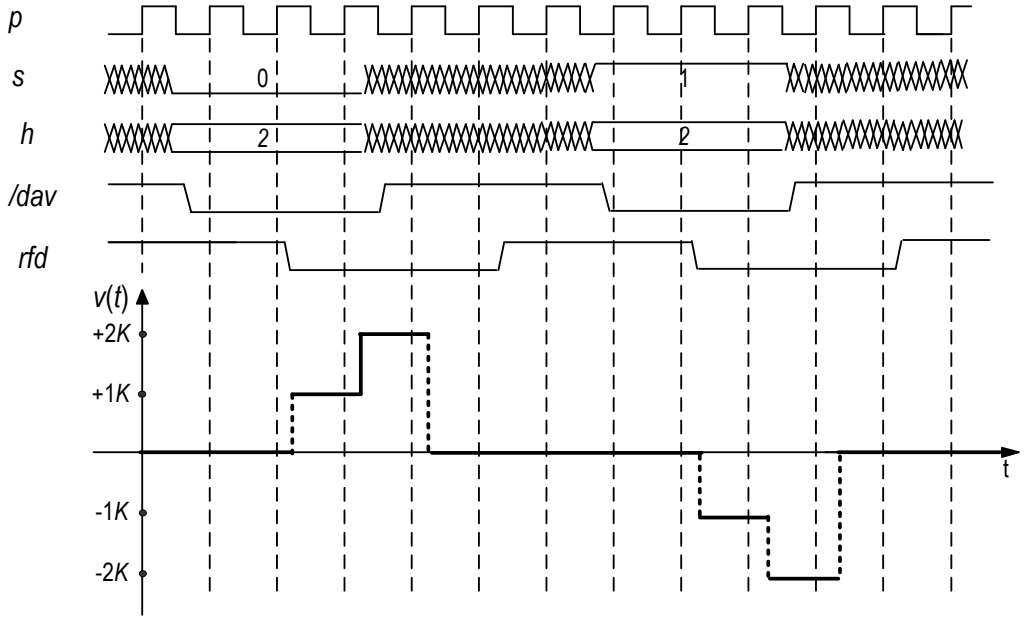


Fig. 2

Completare, come controprova del funzionamento dell'unità descritta, il suo diagramma di temporizzazione

Es 1 - Soluzione

La descrizione del campionatore è la seguente. Si noti che il campionatore necessita di quattro stati interni (invece che tre, come nel FF-D-ET), in quanto deve discriminare “conserva a 0” (S0) da “conserva ad 1” (S3).

t	p=0		p=1		sr
	0	1	0	1	
S0	(S0)	(S0)	S2	S1	0-
S1	S3	S3	(S1)	(S1)	10
S2	S0	S0	(S2)	(S2)	01
S3	(S3)	(S3)	S1	S2	-0

Dalla tabella di flusso si osserva che la rete è normale, e soggetta ad alee essenziali. Adottando la codifica S0=00, S1=01, S2=10, S3=11 non si hanno corse delle variabili di stato. Per quanto riguarda la sintesi di CN1 si può subito dire che $t_{mark} = t_{CN1}$ (perché ci sono alee essenziali), e che quindi $T = t_{CN1} + t_{mark} + t_{CN1} = 3t_{CN1}$ è il minimo tempo di permanenza dello stato di ingresso (la rete è normale).

Per quanto riguarda CN1, dalla tabella seguente si ottengono le due seguenti sintesi SP prive di alee del 1° ordine:

$$a_1 = \overline{p} \cdot y_0 + y_1 \cdot y_0 \cdot t + \overline{y_1} \cdot y_0 \cdot p + t \cdot p \cdot y_0 + \bar{t} \cdot p \cdot \overline{y_1}$$

$$a_0 = \overline{y_1} \cdot y_0 + \bar{t} \cdot y_0 + \overline{p} \cdot y_0 + t \cdot p \cdot \overline{y_1}$$

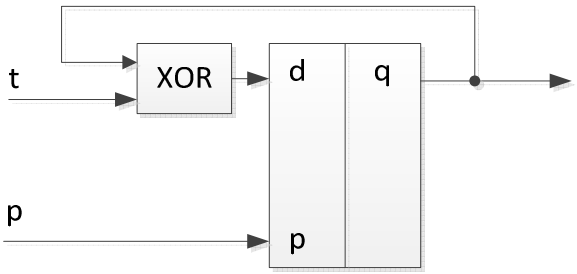
Per quanto riguarda CN2, abbiamo

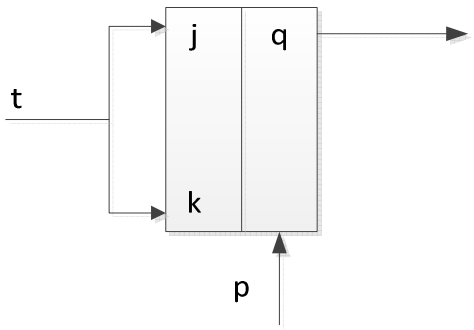
$s = y_0, r = \overline{y_0}$.

tp / y1y0	00	01	11	10
00	00	10	01	00
01	11	01	01	11
11	11	01	10	11
01	00	10	10	00

a1a0

3) Le due sintesi euristiche richieste sono le seguenti:





Esercizio 2 – soluzione

SOLUZIONE PIU' SEMPLICE con XXX che lavora in traslazione

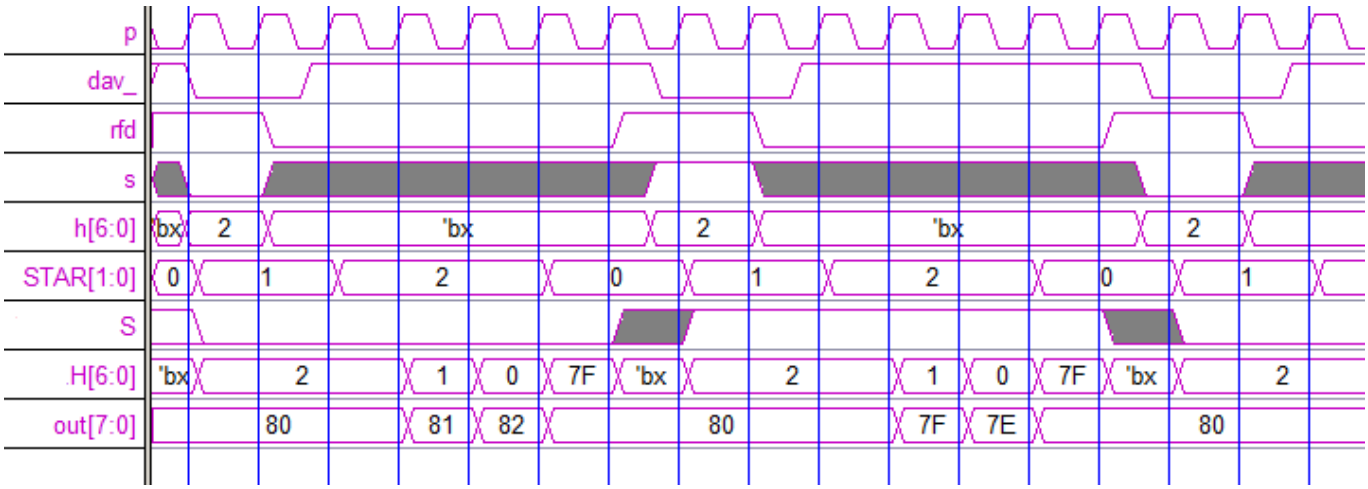
```
module XXX(s,h,rfd,dav_, out, p,reset_);
 input          p,reset_;
 input          dav_;
 output         rfd;
 output[7:0]    out;
 input s; input [6:0] h;

 reg        RFD;    assign rfd=RFD;
 reg [7:0] OUT;     assign out=OUT;

 reg S; reg [6:0] H;

 reg[1:0]  STAR;    parameter S0=0,S1=1,S2=2;

 always @(posedge p or negedge reset_)
  if (reset_==0) begin OUT<='H80; RFD<=1; STAR<=S0; end else #3
  casex(STAR)
   S0: begin RFD<=1; S<=s; H<=h; STAR<=(dav_==1)?S0:S1; end
   S1: begin RFD<=0; STAR<=(dav_==0)?S1:S2; end
   S2: begin H<=H-1; OUT<=(H==0)?'H80:((S==0)?(OUT+1):(OUT-1));
             STAR<=(H==0)?S0:S2; end
  endcase
endmodule
```



SOLUZIONE PIU' SEMPLICE con XXX che lavora in complemento a due

```
module XXX(s,h,rfd,dav_, out, p,reset_);
 input          p,reset_;
 input          dav_;
 output         rfd;
 output[7:0]    out;
 input s; input [6:0] h;

 reg        RFD;    assign rfd=RFD;
 reg [7:0] OUT;     assign out={~OUT[7],OUT[6:0]};

 reg S; reg [6:0] H;

 reg[1:0]  STAR;    parameter S0=0,S1=1,S2=2;

 always @(posedge p or negedge reset_)
  if (reset_==0) begin OUT<='H00; RFD<=1; STAR<=S0; end else #3
  casex(STAR)
   S0: begin RFD<=1; S<=s; H<=h; STAR<=(dav_==1)?S0:S1; end
   S1: begin RFD<=0; STAR<=(dav_==0)?S1:S2; end
   S2: begin H<=H-1; OUT<=(H==0)?'H00:((S==0)?(OUT+1):(OUT-1));
             STAR<=(H==0)?S0:S2; end
  endcase
endmodule
```

Altra soluzione in cui XXX lavora in traslazione

```
module XXX(s,h,rfd,dav_, out, p,reset_);
 input         p,reset_;
 input         dav_;
 output        rfd;
 output[7:0]   out;
 input s; input [6:0] h;
 reg       RFD;    assign rfd=RFD;
 reg [7:0] OUT;    assign out=OUT;
 reg S; reg [6:0] H;
 reg[1:0]  STAR;   parameter S0=0,S1=1,S2=2;
 wire stop=(ABS({~OUT[7],OUT[6:0]})=={1'B0,H})?1:0; //Qui e' il PUNTO
CRUCIALE
 // ABS() e' solita rete che calcola il valore assoluto di un numero
 // rappresentato in complement a due
   function [7:0] ABS;
    input[7:0] A;
    ABS=(A[7]==0)?A:((~A)+1);
   endfunction
  always @(posedge p or negedge reset_)
  if (reset_==0) begin OUT<='H80; RFD<=1; STAR<=S0; end else #3
  casex(STAR)
   S0: begin RFD<=1; S<=s; H<=h; STAR<=(dav_==1)?S0:S1; end
   S1: begin RFD<=0; STAR<=(dav_==0)?S1:S2; end
   S2: begin OUT<=(stop==1)?'H80:((S==0)?(OUT+1):(OUT-1));
             STAR<=(stop==1)?S0:S2; end
  endcase
endmodule
```

Altra soluzione in cui XXX lavora in complemento a due

```
module XXX(s,h,rfd,dav_, out, p,reset_);
 input         p,reset_;
 input         dav_;
 output        rfd;
 output[7:0]   out;
 input s; input [6:0] h;
 reg       RFD;    assign rfd=RFD;
 reg [7:0] OUT;    assign out={~OUT[7],OUT[6:0]};
 reg S; reg [6:0] H;
 reg[1:0]  STAR;   parameter S0=0,S1=1,S2=2;
 wire stop=(ABS(OUT)=={1'B0,H})?1:0; //Qui e' il PUNTO CRUCIALE
 // ABS() e' solita rete che calcola il valore assoluto di un numero
 // rappresentato in complement a due
   function [7:0] ABS;
    input[7:0] A;
    ABS=(A[7]==0)?A:((~A)+1);
   endfunction
  always @(posedge p or negedge reset_)
  if (reset_==0) begin OUT<='H00; RFD<=1; STAR<=S0; end else #3
  casex(STAR)
   S0: begin RFD<=1; S<=s; H<=h; STAR<=(dav_==1)?S0:S1; end
   S1: begin RFD<=0; STAR<=(dav_==0)?S1:S2; end
   S2: begin OUT<=(stop==1)?'H00:((S==0)?(OUT+1):(OUT-1));
             STAR<=(stop==1)?S0:S2; end
  endcase
endmodule
```

Una soluzione col trucco, in cui si blocca il Produttore rimandando la messa a 0 di *rfd* per costringerlo a mantenere costante la sua uscita. Si possono così evitare in *XXX* i registri di appoggio S e H.

```
module XXX(s,h,rfd,dav_, out, p,reset_);
 input         p,reset_;
 input         dav_;
 output        rfd;
 output[7:0]   out;
 input s; input [6:0] h;
 reg       RFD;    assign rfd=RFD;
 reg [7:0] OUT;    assign out=OUT;

 reg[1:0]  STAR;   parameter S0=0,S1=1,S2=2;

 wire stop=(ABS({~OUT[7],OUT[6:0]})=={1'B0,h})?1:0; //Qui e' il PUNTO
CRUCIALE
 // ABS() e' solita rete che calcola il valore assoluto di un numero
 // rappresentato in complement a due
   function [7:0] ABS;
    input[7:0] A;
    ABS=(A[7]==0)?A:((~A)+1);
   endfunction
  always @(posedge p or negedge reset_)
  if (reset_==0) begin OUT<='H80; RFD<=1; STAR<=S0; end else #3
  casex(STAR)
   S0: begin RFD<=1; STAR<=(dav_==1)?S0:S1; end
   S1: begin OUT<=(stop==1)?'H80:((s==0)?(OUT+1):(OUT-1));
             STAR<=(stop==1)?S2:S1; end
   S2: begin RFD<=0; STAR<=(dav_==0)?S2:S0; end
  endcase
endmodule
```