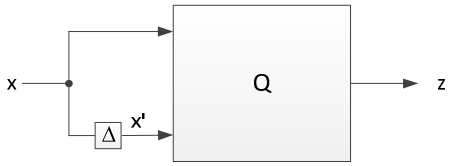
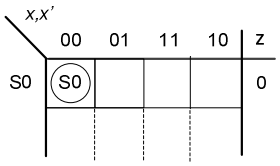
Esercizio 1

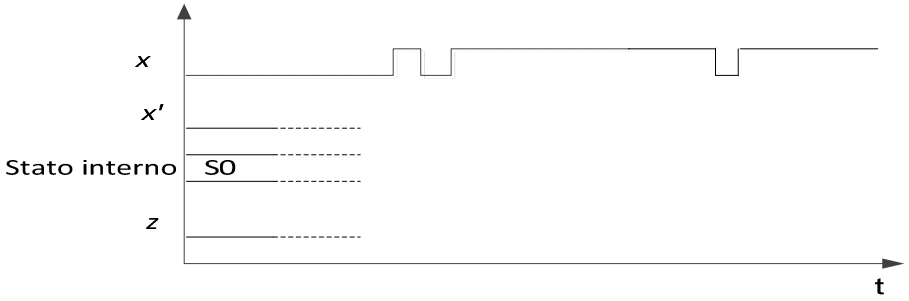
Si consideri una variabile logica *x* soggetta ad alee (sia statiche, su entrambi i livelli, che dinamiche, crescenti e decrescenti). Si sa con certezza che entro un tempo T_{alee} i fenomeni transitori su *x* si sono estinti e che *x* resta a regime per tempi molto maggiori di T_{alee}.

Con riferimento alla seguente figura, descrivere e sintetizzare la rete sequenziale asincrona Q in modo che produca un'uscita *z* che insegua *x* ma sia priva di qualunque tipo di alea.



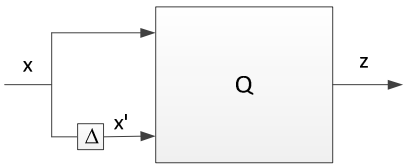


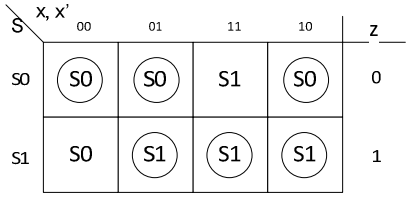
Dimensionare il ritardo Δ e simulare l'evoluzione di Q completando il seguente diagramma temporale:



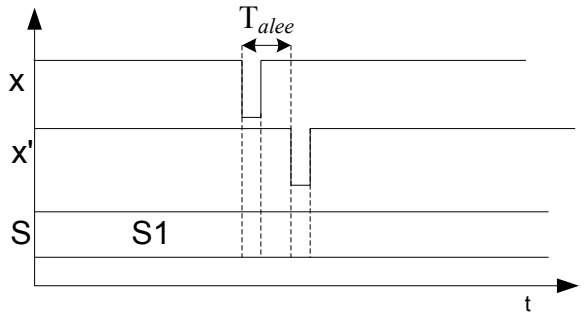
Soluzione

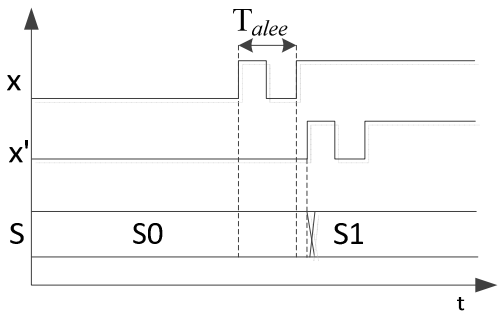
La rete Q ha due stati, corrispondenti ai valori di *z*, e cambia stato soltanto quando *x* e *x'* coincidono.





È immediato constatare che la tabella sopra riportata è priva di alee essenziali. Pertanto, la rete Q può essere implementata con ritardo nullo sulla variabile di stato. Per quest'ultima la codifica può essere assunta uguale all'uscita, quindi CN2 è un cortocircuito. CN1 è la seguente: $a = x \cdot x' + y \cdot x + y \cdot x'$, priva di alee. Per un'alea statica (ad esempio sul livello 1) il diagramma temporale è il seguente:



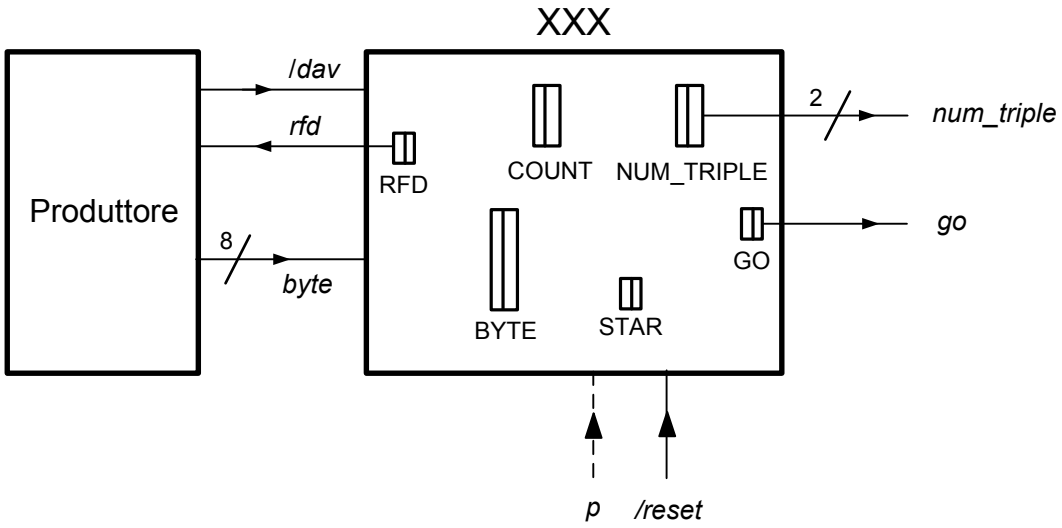


Per un'alea dinamica (ad esempio crescente) il diagramma temporale è il seguente (nel disegno si assume trascurabile il tempo di adeguamento di Q):

Si vede chiaramente che il ritardo da dare a *x'* è $\Delta > T_{alee}$.

Esercizio 2

Descrivere e Sintetizzare l'Unità *XXX* che: 1) Preleva un byte dal Produttore e lo elabora calcolando quante triple di bit 110 sono contenute in esso; 2) Emette il risultato del conteggio tramite la variabile di uscita *num_triple*; 2) Pone ad 1 la variabile di uscita *go* per un periodo del clock.e torna al punto 1



Usare per i fili e per i registri i nome riportati in Figura

Possibile Soluzione

```
//----------------------------------------------------------------------------

module XXX(byte,dav_,rfd, num_triple,go, p,reset_);
  input       p,reset_;

  input [7:0] byte;
  input       dav_;
  output      rfd;
  output[1:0] num_triple;
  output      go;

  reg       RFD;         assign rfd=RFD;
  reg [1:0] NUM_TRIPLE;  assign num_triple=NUM_TRIPLE;
  reg       GO;          assign go=GO;

  reg [7:0] BYTE;
  reg [1:0] COUNT;  //Contatore del numero di triple

  reg [1:0] STAR;
  parameter[1:0] S0=0, S1=1, S2=2, S3=3;

  always @(posedge p or negedge reset_)if (reset_==0) begin RFD<=1; GO<=0;
STAR=S0; end else #3
   casex(STAR)
    S0: begin GO<=0; RFD<=1; BYTE<=byte; STAR<=(dav_==1)?S0:S1; end
    S1: begin COUNT<=0; RFD<=0; STAR<=(dav_==0)?S1:S2; end
    S2: begin COUNT<=(BYTE[7:5]=='B110)?(COUNT+1):COUNT;
              BYTE<=(BYTE[7:5]=='B110)?({BYTE[4:0],3'B111}):({BYTE[6:0],1'B1});
              STAR<=(BYTE=='HFF)?S3:S2; end
    S3: begin NUM_TRIPLE<=COUNT; GO<=1; STAR<=S0; end
 endcase
endmodule

//----------------------------------------------------------------------------
```