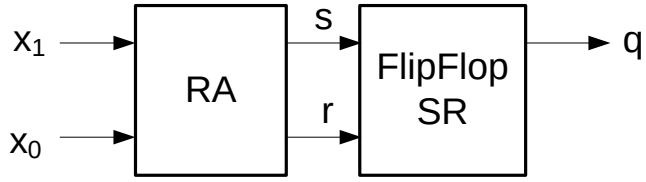
Esercizio 1. Si consideri il seguente sistema



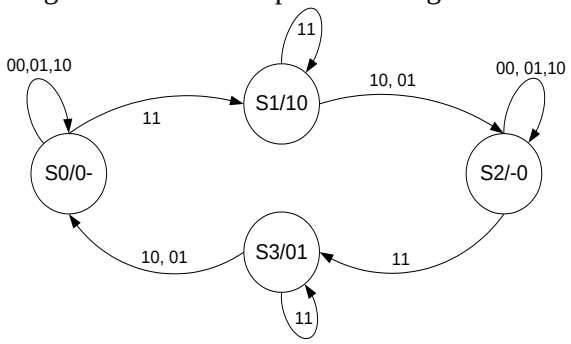
Descrivere e sintetizzare (utilizzando elementi neutri di ritardo come meccanismi di marcatura) la rete sequenziale asincrona RA in modo tale che la variabile q *commuti* ogni qual volta si presenta in ingresso al sistema lo stato $x_1x_0 = 11$. Sintetizzare le reti combinatorie in forma SP, e calcolarne il costo a porte e a diodi.

Soluzione

La rete RA deve essere fatta in modo tale che le sue uscite siano

- alternativamente 10, 01 quando gli ingressi sono 11
- 00 in tutti gli altri casi.

Il diagramma di flusso è pertanto il seguente:



che corrisponde alla seguente tabella di flusso:

x_1x_0	00	01	11	10	sr
S0	(S0)	(S0)	S1	(S0)	0-
S1	--	S2	(S1)	S2	10
S2	(S2)	(S2)	S3	(S2)	-0
S3	--	S0	(S3)	S0	01

Si osservi che la tabella di flusso di cui sopra (che è normale) è soggetta ad alee essenziali, pertanto gli elementi neutri di ritardo dovranno avere un ritardo non inferiore a T_{CN1}.

Calcolare il ritardo degli elementi neutri ed il tempo minimo di permanenza di uno stato di ingresso.

NOTE

1 Si ricordi che RA è una rete sequenziale asincrona e quindi, quando riceve in ingresso $x_1x_0 = 11$, compie un passo e poi si stabilizza per tutto il tempo in cui lo stato di ingresso permane.
2 Non ci si preoccupi che il tutto risponda alle specifiche fin dall'arrivo del primo stato di ingesso $x_1x_0 = 11$ immediatamente successivo all'accensione.

Adottando la codifica S0=00, S1=10, S2=11, S3=01, si ottiene per la rete RC_Z l'espressione $s = y_1$, $r = \overline{y_1}$. Pertanto, il costo di RC_Z è nullo.

Utilizzando come meccanismo di marcatura degli elementi neutri di ritardo, si ottengono le seguenti mappe per la rete combinatoria RC_A:

y_1y_0 \ x_1x_0	00	01	11	10
00	0	0	1	0
01	--	0	0	0
11	1	1	0	1
10	--	1	1	1

a1

y_1y_0 \ x_1x_0	00	01	11	10
00	0	0	0	0
01	--	0	1	0
11	1	1	1	1
10	--	1	0	1

a0

Dalle quali si ottiene:

$$a_1 = y_1 \cdot \overline{x_1} + y_1 \cdot \overline{x_0} + x_1 \cdot x_0 \cdot \overline{y_0} + y_1 \cdot \overline{y_0},$$
$$a_0 = y_1 \cdot \overline{x_1} + y_1 \cdot \overline{x_0} + x_1 \cdot x_0 \cdot y_0 + y_1 \cdot y_0.$$

In entrambi i casi, l'ultimo implicante è ridondante, ed è aggiunto per evitare alee del primo ordine..

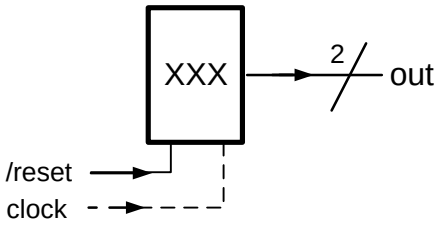
Il costo a porte della rete RC_A è 8 (e non 10), in quanto le stesse due porte AND possono essere utilizzate contemporaneamente nella sintesi di a_1 ed a_0. Analogamente, il costo a diodi è 22.

Visto che la rete è normale, il minimo tempo di permanenza di uno stato di ingresso dovrà essere $3T_{CN1}$.

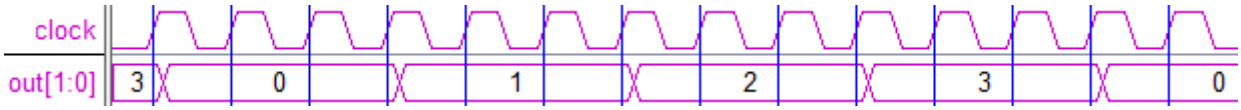
Esercizio 2

Descrivere e sintetizzare la Rete Sequenziale Sincronizzata XXX che, partendo al reset con *out* a 3, si evolve all'infinito come segue come segue:

a) Mette *out* a 0 e lo tiene per **M** periodi di clock
b) Mette *out* a 1 e lo tiene per **M** periodi di clock
c) Mette *out* a 2 e lo tiene per **M** periodi di clock
d) Mette *out* a 3 e lo tiene per **M** periodi di clock



ovvero, per **M=3:**



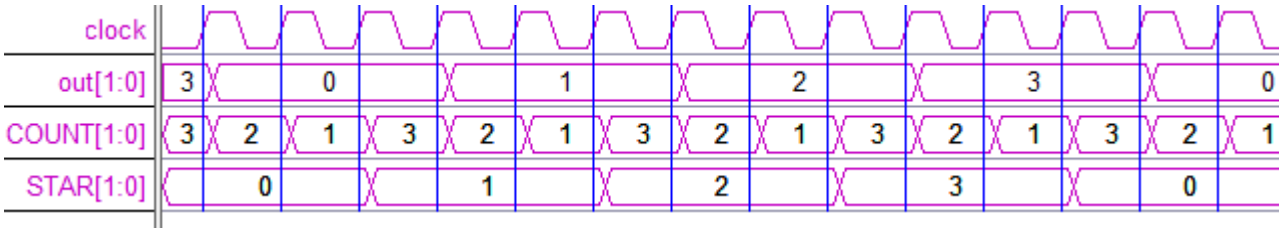
Nella descrizione dichiarare:

```
reg[…:0] COUNT; parameter M=….;
```

Partendo dalla descrizione Verilog, si tracci quindi l'evoluzione di XXX per **M**=3, verificando che rispetti la temporizzazione di cui sopra (*Data la semplicità dell'Unità è ESSENZIALE che essa risponda esattamente alla temporizzazione richiesta*).

Soluzione più diluita

```
module XXX(out,clock,reset_);
 input   clock,reset_;
 output  [1:0] out;
 reg     [1:0] OUT; assign out=OUT;
 reg     [1:0] COUNT; parameter M=3 ;      // reg [...:0] COUNT; parameter M=…
 reg     [1:0] STAR;  parameter S0=0,S1=1,S2=2,S3=3;
always @( reset_==0) begin COUNT=M; OUT<=3; STAR=S0; end
 always @(posedge clock) if (reset_==1) #3
  casex(STAR)
   S0: begin OUT<=0; COUNT<=(COUNT==1)?M:(COUNT-1); STAR<=(COUNT==1)?S1:S0; end
   S1: begin OUT<=1; COUNT<=(COUNT==1)?M:(COUNT-1); STAR<=(COUNT==1)?S2:S1; end
   S2: begin OUT<=2; COUNT<=(COUNT==1)?M:(COUNT-1); STAR<=(COUNT==1)?S3:S2; end
   S3: begin OUT<=3; COUNT<=(COUNT==1)?M:(COUNT-1); STAR<=(COUNT==1)?S0:S3; end
  endcase
endmodule
```



Soluzione ottimizzata

```
module XXX(out,clock,reset_);
input   clock,reset_;
output  [1:0] out;
reg     [1:0] OUT; assign out=OUT;
reg     [1:0] COUNT; parameter M=3 ;
always @( reset_==0) begin COUNT=M; OUT<='B11; end
always @(posedge clock) if (reset_==1) #3
   begin OUT<=(COUNT==3)?(OUT+1):OUT; COUNT<=(COUNT==1)?M:(COUNT-1); end
endmodule
```