Esercizio 1

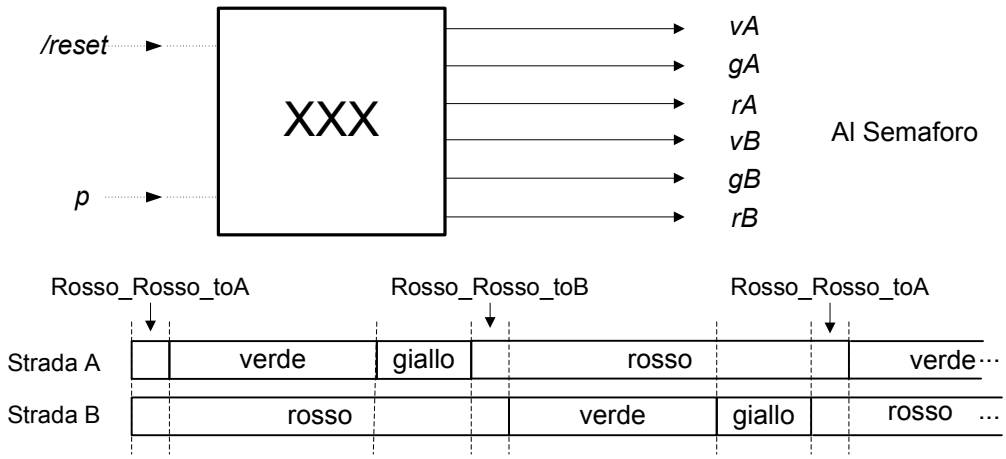
Descrivere e sintetizzare (secondo il modello a elementi neutri di ritardo) un *verificatore di handshake*. Tale rete è una rete sequenziale asincrona con due ingressi `dav_` e `rfd` e un'uscita `error`, che analizza se gli handshake tra due altre reti avvengono correttamente. Si comporta come segue:

- Fintanto che gli handshake si stanno svolgendo in maniera corretta, l'uscita vale zero.
- Non appena le variabili di ingresso hanno una transizione non consentita dalle regole dell'handshake, la rete porta l'uscita ad 1 e *resta bloccata in eterno*.

Si assuma che al reset gli ingressi valgano 11.

Esercizio 2

Descrivere l'Unità XXX che comanda un semaforo all'incrocio di due strade A e B in accordo alle specifiche riportate nella figura (*vA* a 1 accende il verde nel percorso relativo alla strada A, *gA* a 1 accende il giallo nel percorso relativo alla strada A, *rA* a 1 accende il rosso nel percorso relativo alla strada A, …).



Nota:

1) Far durare il verde per 20 cicli di clock, il giallo per 3 cicli di clock ed il rosso-rosso per 1 ciclo di clock
2) Inizializzare l'Unità XXX nello stato interno *S*0 indicante *Rosso_Rosso_toA*:

ovvero

```
module XXX(vA,gA,rA, vB,gB,rB, p,reset_);
 input       p,reset_;
 output      vA,gA,rA, vB,gB,rB;
 reg         VA,GA,RA, VB,GB,RB; assign vA=VA,gA=GA,rA=RA, vB=VB,gB=GB,rB=RB;
 reg [...:0] COUNT;
 reg [2:0]   STAR; parameter  S0=0, S1=1, ... ;
 always @(posedge p or negedge reset_)
  if (reset_==0) begin VA=0; VB=0; GA=0; GB=0; RA=1; RB=1; STAR=S0; end else #3
   casex(STAR)
    S0:  ...
    ...
  endcase
endmodule
```

<u>Fondamentale</u>: Simulare l'evoluzione dell'Unità da voi descritta ammettendo, per brevità, che il verde duri 4 cicli di clock ed il giallo 2 cicli di clock.

ES 1 – Una possibile soluzione

La tabella di flusso della rete è la seguente:

dav_ rfd	00	01	11	10	error
S0	--	S1	(S0)	Serr	0
S1	S2	(S1)	Serr	--	0
S2	(S2)	Serr	--	S3	0
S3	Serr	--	S0	(S3)	0
Serr	(Serr)	(Serr)	(Serr)	(Serr)	1

La rete presenta alee essenziali, e quindi sono necessari elementi di ritardo. Inoltre, essendo cinque gli stati interni, sono necessarie tre variabili di stato. Lo stato `Serr` deve essere adiacente a tutti gli altri. Per evitare corse delle variabili di stato, conviene adottare la seguente codifica:

```
S0: 000 S1: 001, S2: 011, S3: 010,
Serr: 1xx
```

In tal modo, lo stato `Serr` viene suddiviso in quattro stati, in ciascuno dei quali l'uscita vale 1, e non si hanno corse delle variabili di stato.
Secondo una sintesi basata su un modello con elementi neutri di ritardo, le tabelle di verità conseguenti alla codifica di cui sopra sono le seguenti:

y_1y_0 \ dav_ rfd	00	01	11	10
00	---	001	000	100
01	011	001	101	---
11	011	111	---	010
10	110	---	000	010

$a_2a_1a_0$, y_2=0

y_1y_0 \ dav_ rfd	00	01	11	10
00	1--	1--	1--	1--
01	1--	1--	1--	1--
11	1--	1--	1--	1--
10	1--	1--	1--	1--

$a_2a_1a_0$, y_2=1

Dalle tabelle si ricava quanto segue:

$$error = y_2$$

$$a_2 = y_2 + dav\_ \cdot \overline{rfd} \cdot \overline{y_1} + \overline{dav\_} \cdot rfd \cdot y_1 + dav\_ \cdot rfd \cdot y_0 + \overline{dav\_} \cdot \overline{rfd} \cdot \overline{y_0}$$

$$a_1 = y_1 \cdot y_0 + \overline{dav\_} \cdot \overline{rfd} + y_1 \cdot \overline{rfd}$$

$$a_0 = \overline{y_1} \cdot y_0 + \overline{dav\_} \cdot rfd + y_0 \cdot \overline{dav\_}$$

ES 2 - Una possibile Soluzione

```
module XXX(vA,gA,rA, vB,gB,rB, p,reset_);
 input      p,reset_;
 output     vA,gA,rA, vB,gB,rB;
 reg        VA,GA,RA, VB,GB,RB; assign vA=VA,gA=GA,rA=RA, vB=VB,gB=GB,rB=RB;
 reg [4:0]  COUNT;
 reg [2:0]  STAR;
 parameter  S0=0, S1=1, S2=2, S3=3, S4=4, S5=5;
 parameter  durata_verde=20, durata_giallo=3;
 always @(posedge p or negedge reset_)
  if (reset_==0) begin VA=0;VB=0;GA=0;GB=0;RA=1;RB=1;STAR=S0; end
  else #15
   casex(STAR)
    S0: begin VA<=0; GA<=0; RA<=1; VB<=0; GB<=0; RB<=1; COUNT<=durata_verde;
              STAR<=S1; end
    S1: begin VA<=1; RA<=0; GA<=0; COUNT<=(COUNT==1)?durata_giallo:(COUNT-1);
              STAR<=(COUNT==1)?S2:S1; end
    S2: begin VA<=0; RA<=0; GA<=1; COUNT<=COUNT-1;
              STAR<=(COUNT==1)?S3:S2; end
    S3: begin VA<=0; GA<=0; RA<=1; VB<=0; GB<=0; RB<=1; COUNT<=durata_verde;
              STAR<=S4; end
    S4: begin VB<=1; RB<=0; GB<=0; COUNT<=(COUNT==1)?durata_giallo:(COUNT-1);
        STAR<=(COUNT==1)?S5:S4; end
    S5: begin VB<=0; RB<=0; GB<=1; COUNT<=COUNT-1;
        STAR<=(COUNT==1)?S0:S5; end
  endcase
endmodule
```

Temporizzazione nel caso richiesto