DEI
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sistemi Digitali

Esercizi su logica sequenziale

Marta Bagatin, marta.bagatin@unipd.it

Corso di Laurea in Ingegneria dell'Informazione

Anno accademico 2022-2023

Esercizio 4.6

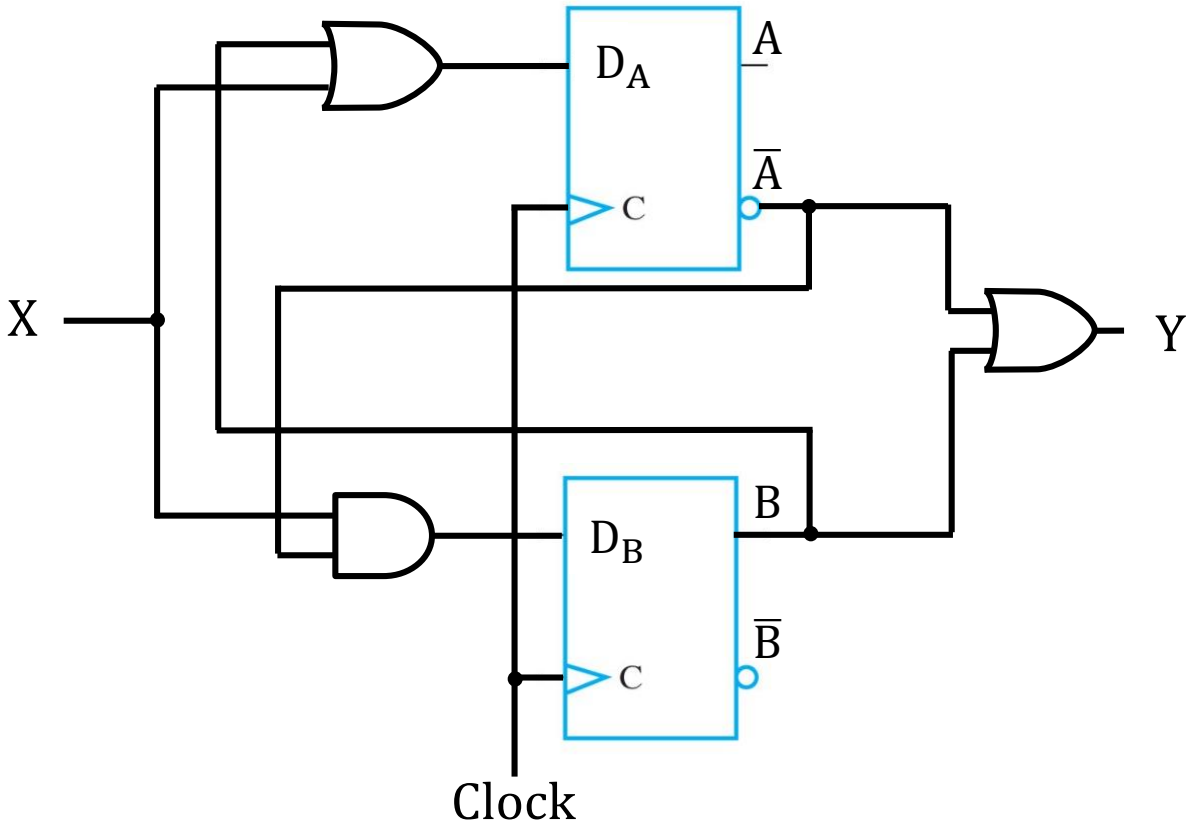
- Esempio di **analisi di un circuito sequenziale**, a partire dalle equazioni di ingresso dei flip-flop e l'equazione dell'uscita
- Un circuito sequenziale con due D flip-flop A e B, ingresso X e uscita Y è specificato dalle seguenti equazioni:
  - $Y = \overline{A} + B$
  - $D_A = X + B$
  - $D_B = X \cdot \overline{A}$
  - Reset porta il sistema nello stato iniziale «00»

a) Disegnare il diagramma logico del circuito
b) Derivare la tabella degli stati
c) Specificare se si tratta di una macchina di Mealy o di Moore
d) Derivare il diagramma degli stati

Esercizio 4.6

a) Disegnare il **diagramma logico** del circuito

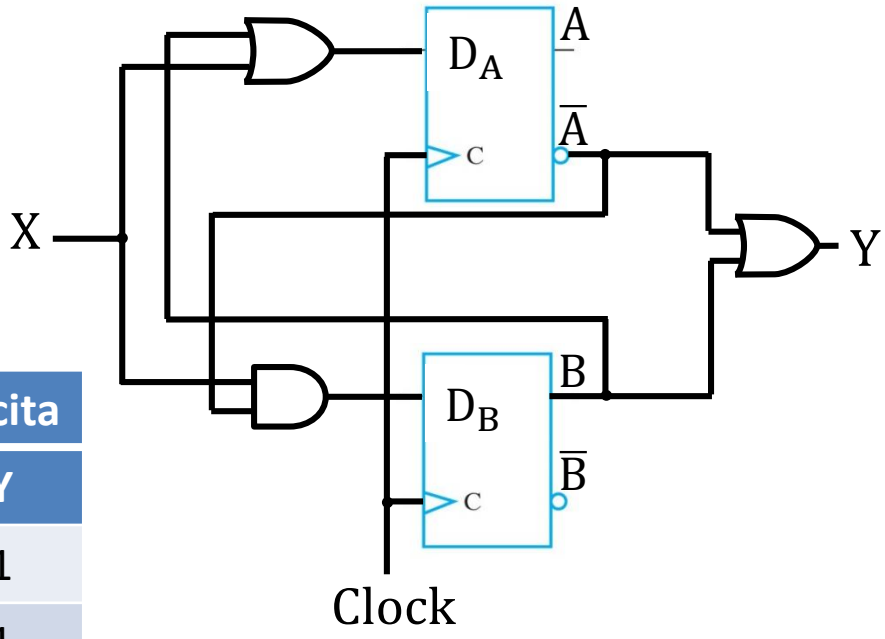
- $Y = \overline{A} + B$
- $D_A = X + B$
- $D_B = X \cdot \overline{A}$



Esercizio 4.6

b) Derivare la **tabella degli stati**

- $Y = \overline{A} + B$
- $D_A = X + B$
- $D_B = X \cdot \overline{A}$



Stato presente		Ingresso	Stato futuro		Uscita
A	B	X	D_A	D_B	Y
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	0	1

Esercizio 4.6

c) E' una macchina di Mealy o di Moore? Di **Moore**, perchè l'uscita dipende SOLO dallo stato presente e non dall'ingresso

- $Y = \overline{A} + B$
- $D_A = X + B$
- $D_B = X \cdot \overline{A}$

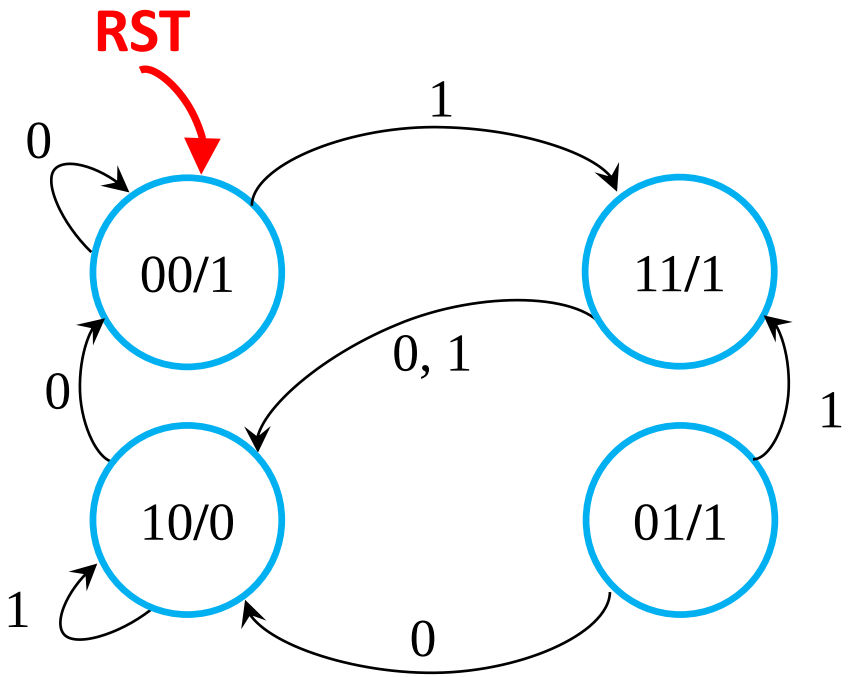
Stato presente		Ingresso	Stato futuro		Uscita
A	B	X	D_A	D_B	Y
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	0	1

Esercizio 4.6

d) Trovare il **diagramma degli stati**

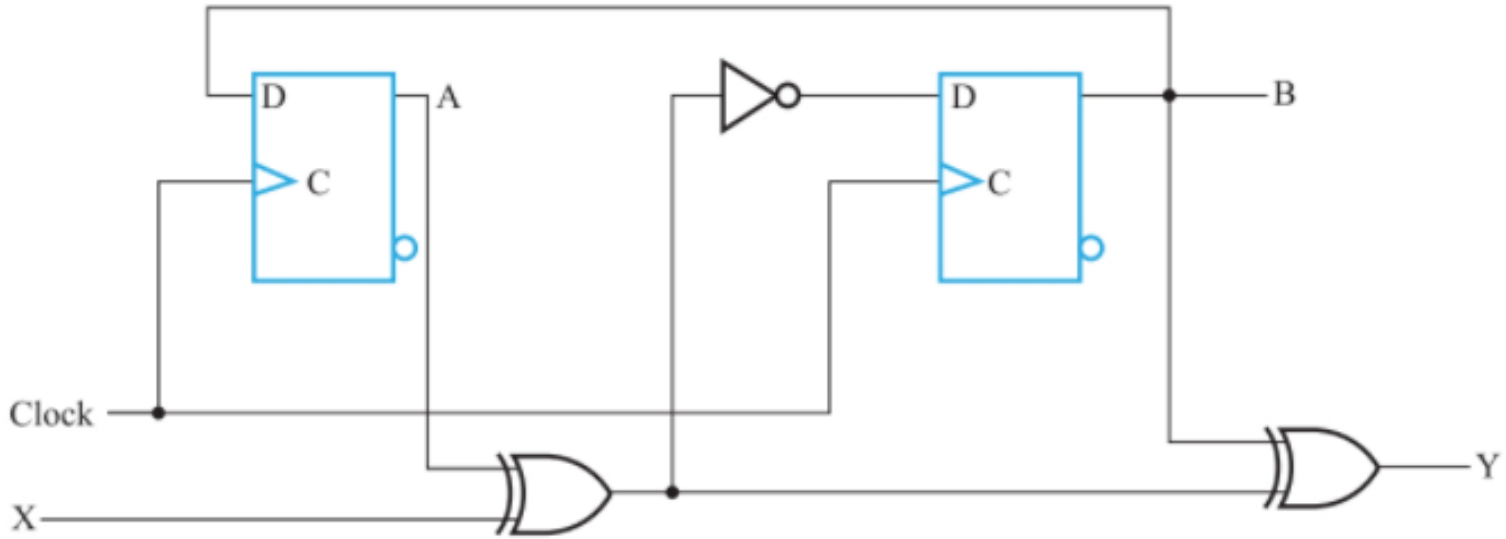
- $Y = \overline{A} + B$
- $D_A = X + B$
- $D_B = X \cdot \overline{A}$

Stato presente		Ingresso	Stato futuro		Uscita
A	B	X	D_A	D_B	Y
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	0	1



Esercizio 4.11

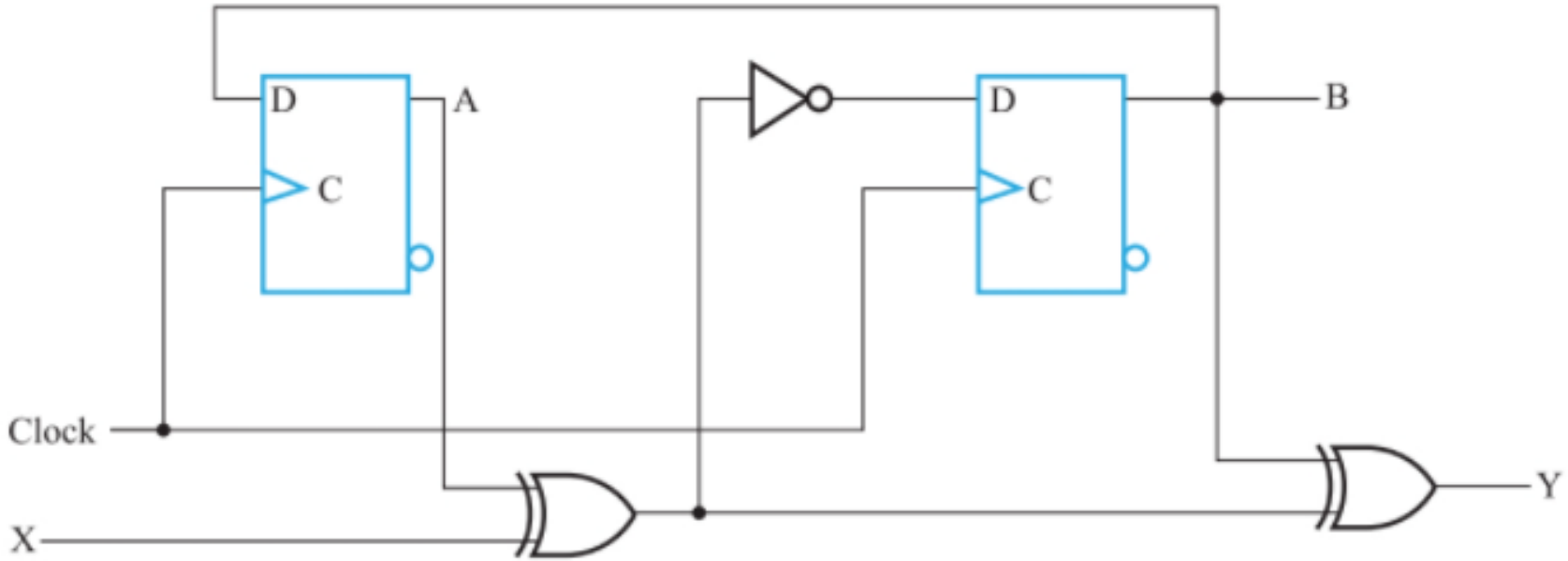
- Esempio di **analisi di un circuito sequenziale**, a partire dal diagramma logico
- Un circuito sequenziale è costituito da due D flip-flop (con reset asincrono), un ingresso X e un'uscita Y. Reset porta il sistema nello stato iniziale «00». Il diagramma logico è mostrato in figura:



a) Derivare le equazioni di aggiornamento di stato e dell'uscita
b) Trovare la tabella degli stati e dell'uscita
c) Derivare il diagramma degli stati

Esercizio 4.11

a) Derivare le equazioni di aggiornamento di stato e dell'uscita



Equazioni di aggiornamento di stato: $D_A = A\,(t+1) = B$

$$D_B = B\,(t+1) = \overline{X \oplus A}$$

Equazione dell'uscita: $Y = B \oplus X \oplus A$

Esercizio 4.11

b) Derivare la tabella degli stati e dell'uscita

Equazioni di aggiornamento di stato: $D_A = A\,(t+1) = B$

$D_B = B\,(t+1) = \overline{X \oplus A}$

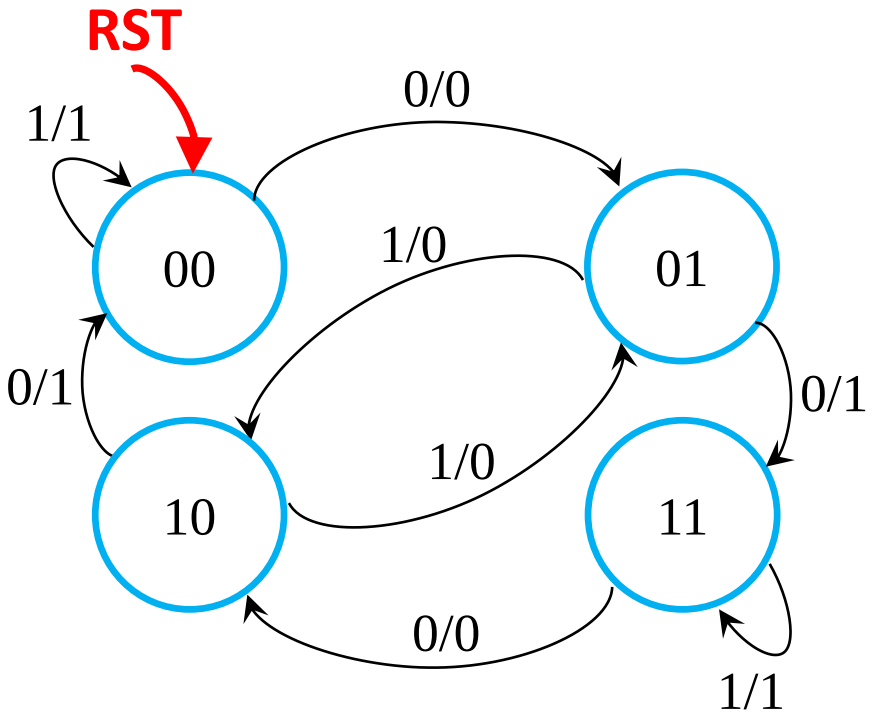
Equazione dell'uscita: $Y = B \oplus (X \oplus A)$

Stato presente		Ingresso	Stato futuro		Uscita
A	B	X	D_A	D_B	Z
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	1

Esercizio 4.11

c) Derivare il diagramma degli stati

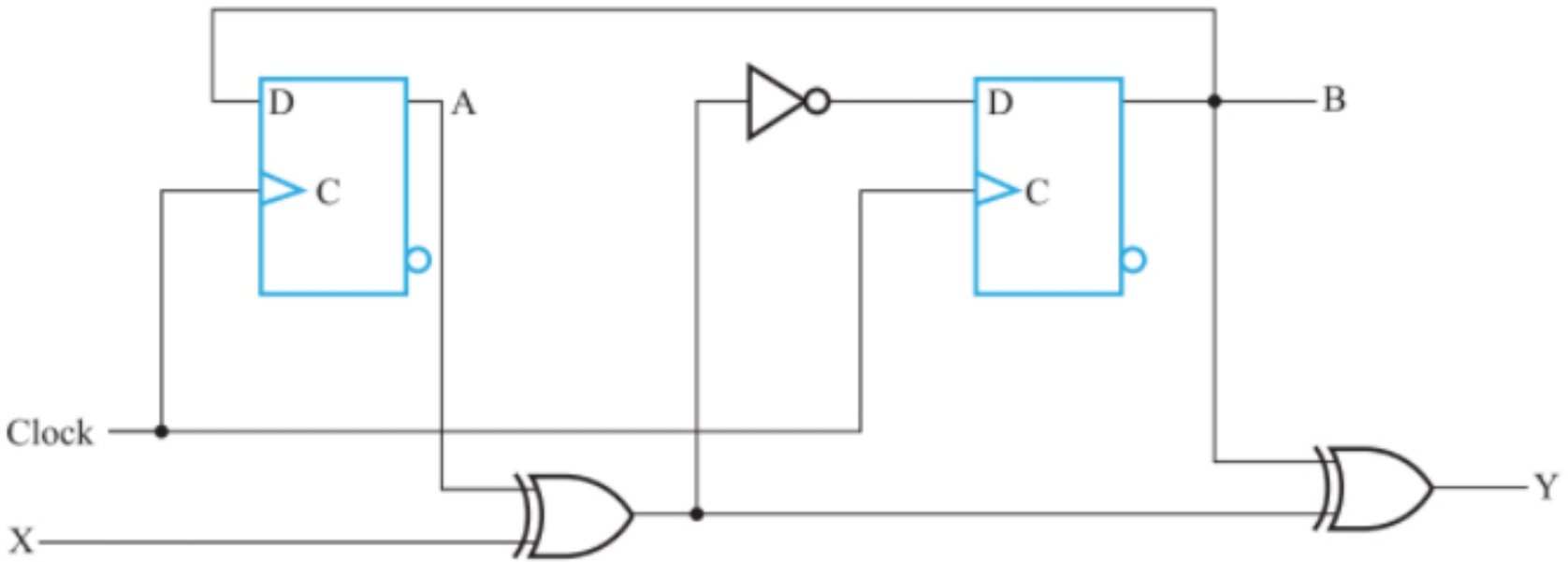
Stato presente		Ingresso	Stato futuro		Uscita
A	B	X	D_A	D_B	Z
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	1



Modello di Mealy
(l'uscita dipende sia dallo stato che dall'ingresso)

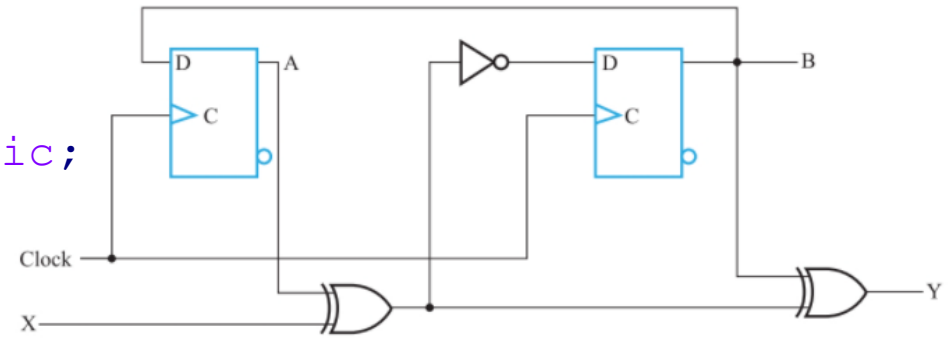
Esercizio 4.40

- Fornire una descrizione **VHDL structural** del circuito nell'es. 4.11:



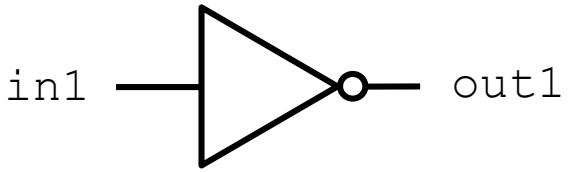
Esercizio 4.40

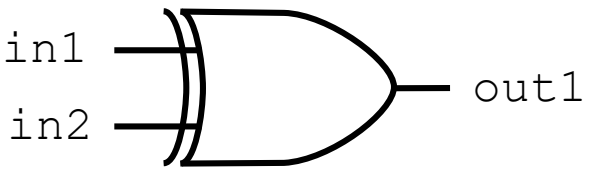
```
library IEEE;
use IEEE.std_logic_1164.all;

entity prob_4_40 is
   port ( clk, rst, X: in std_logic;
                  Y: out std_logic);
end prob_4_40;

architecture impl_struct of prob_4_40 is

   signal A, B, XxorA, XxorA_n: std_logic;

-- istanziazione diretta dei componenti
begin
   INV: entity work.inv(impl) port map (XxorA, XxorA_n);
   DFFA: entity work.dff(impl) port map (rst, clk, B, A);
   DFFB: entity work.dff(impl) port map (rst, clk, XxorA_n, B);
   XOR1: entity work.xor2(impl) port map (A, X, XxorA);
   XOR2: entity work.xor2(impl) port map (B, XxorA, Y);

end impl_struct;
```



Esercizio 4.40

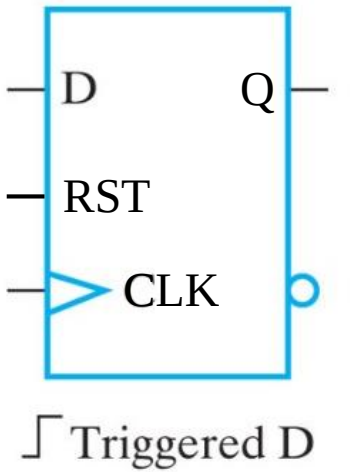
```
-- descrizione delle porte logiche
library ieee;
use ieee.std_logic_1164.all;
entity inverter is
   port(in1 : in std_logic;
        out1 : out std_logic);
end inverter;
architecture impl of inverter is
begin
   out1 <= not in1;
end impl;

library ieee;
use ieee.std_logic_1164.all;
entity xor2 is
   port(in1, in2 : in std_logic;
             out1 : out std_logic);
end xor2;
architecture impl of xor2 is
begin
   out1 <= in1 xor in2;
end impl;
```





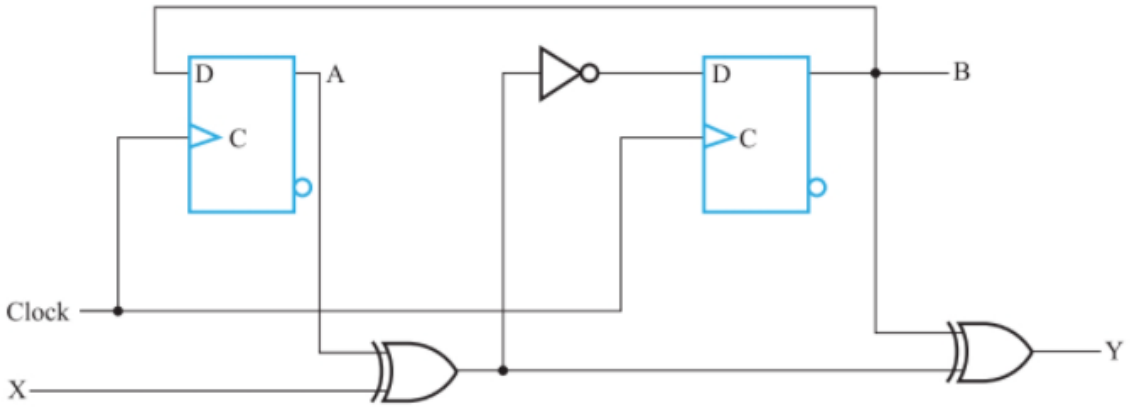
Esercizio 4.40

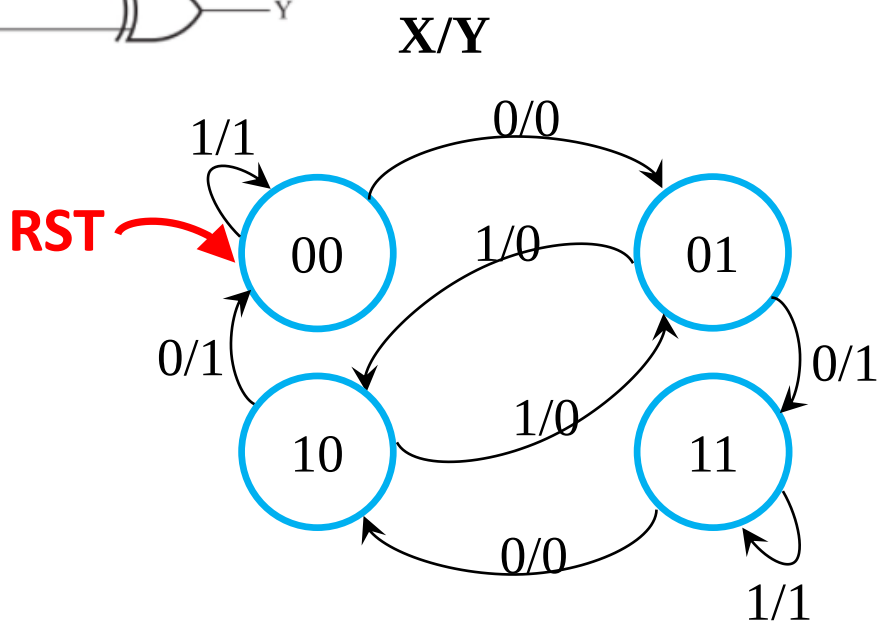
```
-- descrizione del flip-flop D con reset asincrono

library ieee;
use ieee.std_logic_1164.all;
entity dff is
       port(rst, clk, d : in std_logic;
                          q : out std_logic);
end dff;

architecture impl of dff is
begin
process (rst, clk)
    begin
       if rst = '1' then
              q <= '0';
       elsif (clk'event and clk <= '1') then
              q <= d;
       end if;
end process;
end impl;
```



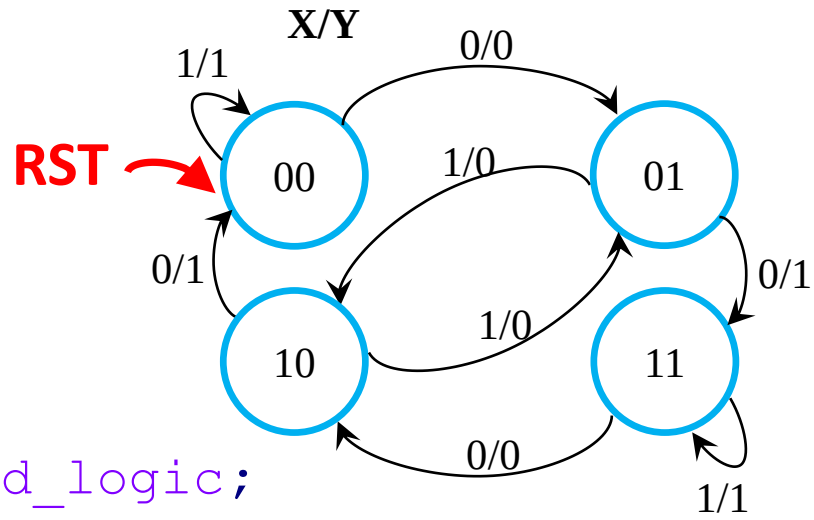
Esercizio 4.41

- Fornire una descrizione **VHDL behavioral** del circuito nell'es. 4.11, usando un process per descrivere il diagramma degli stati:



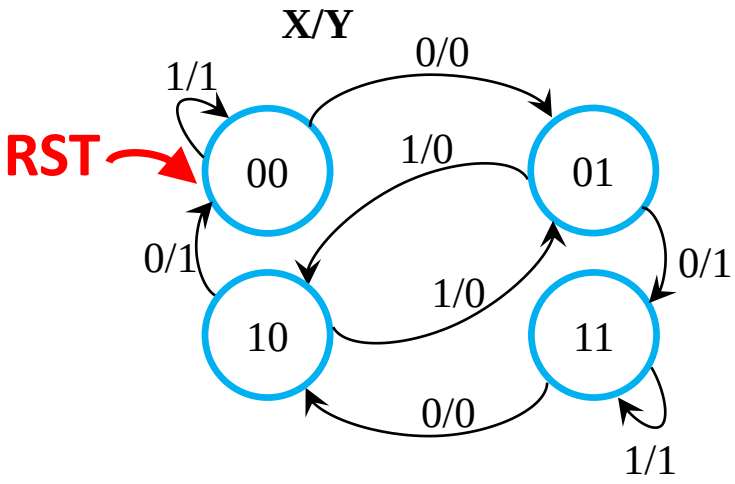


Esercizio 4.41



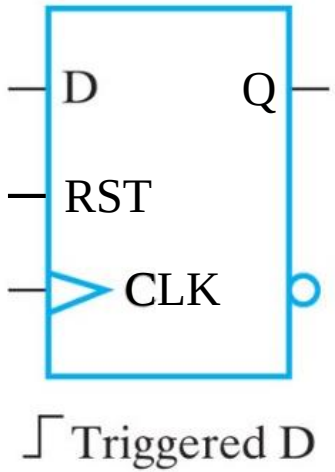

```
library IEEE;
use IEEE.std_logic_1164.all;

entity prob_4_41 is
      port ( clk, rst, X: in std_logic;
                       Y: out std_logic);
end prob_4_41;

architecture impl_beh of prob_4_41 is

    type state_type is (S0, S1, S2, S3);
    signal state, next_state: state_type;

begin
```

Esercizio 4.41



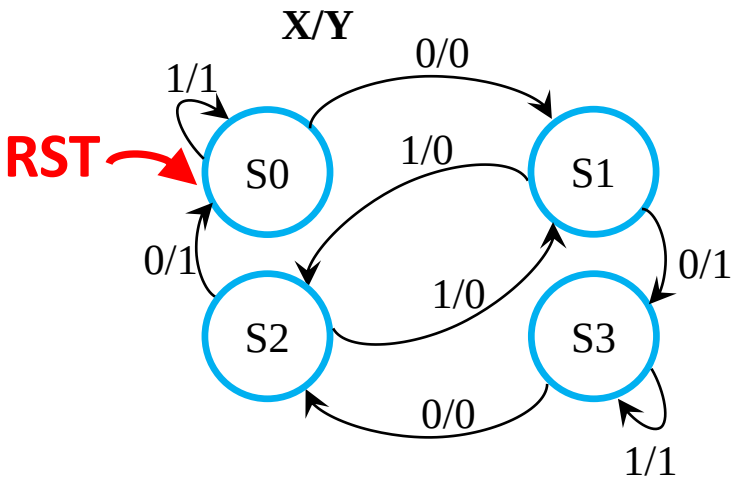

```
-- 1° process: aggiorna lo stato e
implementa il reset asincrono

state_register: process (clk, rst)
begin
     if rst then
        state <= S0;
     elsif (clk'event and clk = '1') then
        state <= next_state;
     end if;
end process;
```

Processo 1: Realizza il registro di stato con D-FF positive edge triggered



Esercizio 4.41

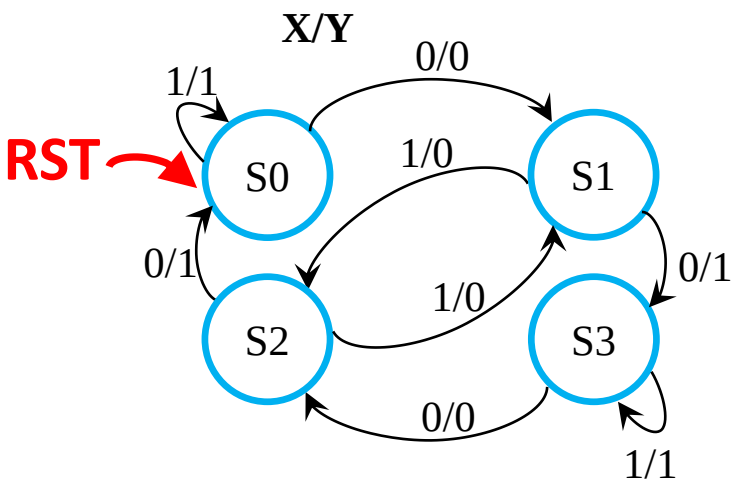
```
-- 2° process: calcola lo stato futuro come funzione
combinatoria di stato presente e ingresso

next_state_comb: process (X, state)
begin
    case state is
        when S0 =>
            if X = '1' then
                next_state <= S0;
            else
                next_state <= S1;
            end if;

        when S1 =>
            if X = '1' then
                next_state <= S2;
            else
                next_state <= S3;
            end if;
```



Processo 2: Realizza la logica combinatoria per calcolare lo stato futuro in funzione di ingresso e stato presente

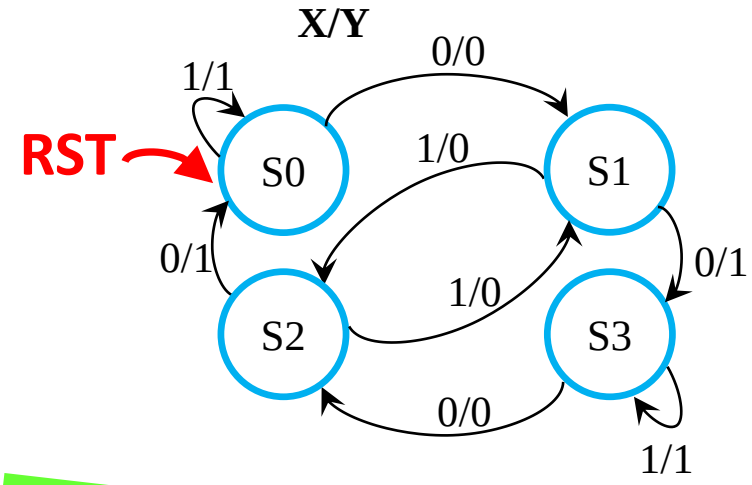
Esercizio 4.41

```
-- 2° process: continued

        when S2 =>
            if X = '1' then
                next_state <= S1;
            else
                next_state <= S0;
            end if;

        when S3 =>
            if X = '1' then
                next_state <= S3;
            else
                next_state <= S2;
            end if;
    end case;
end process;
```



Esercizio 4.41

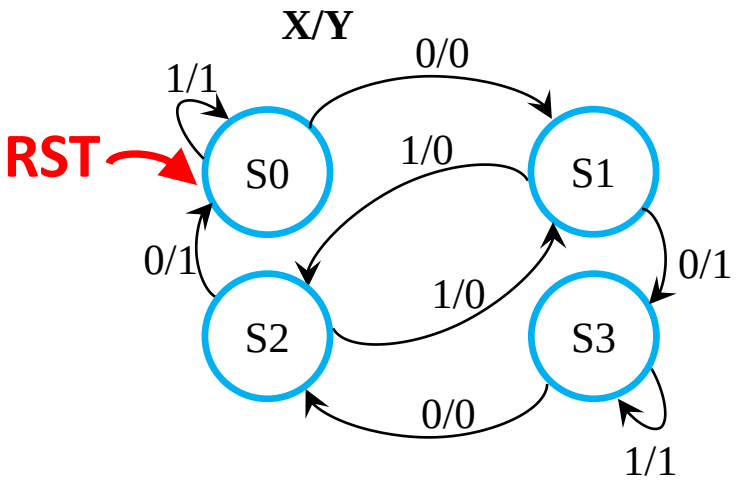
```
-- 3° process: calcola l'uscita come funzione combinatoria
di stato presente e ingresso

output_comb: process (X, state)
begin
    case state is
        when S0 =>
            if X = '1' then
                Y <= '1';
            else
                Y <= '0';
            end if;

        when S1 =>
            if X = '0' then
                Y <= '1';
            else
                Y <= '0';
            end if;
```



Processo 3: Realizza la logica combinatoria per calcolare l'uscita in funzione di ingresso e stato presente

Esercizio 4.41

```
-- 3° process: continued

        when S2 =>
            if X = '0' then
                Y <= '1';
            else
                Y <= '0';
            end if;

        when S3 =>
            if X = '1' then
                Y <= '1';
            else
                Y <= '0';
            end if;
    end case;
end process;

end impl_beh;
```



Esercizio 4.26

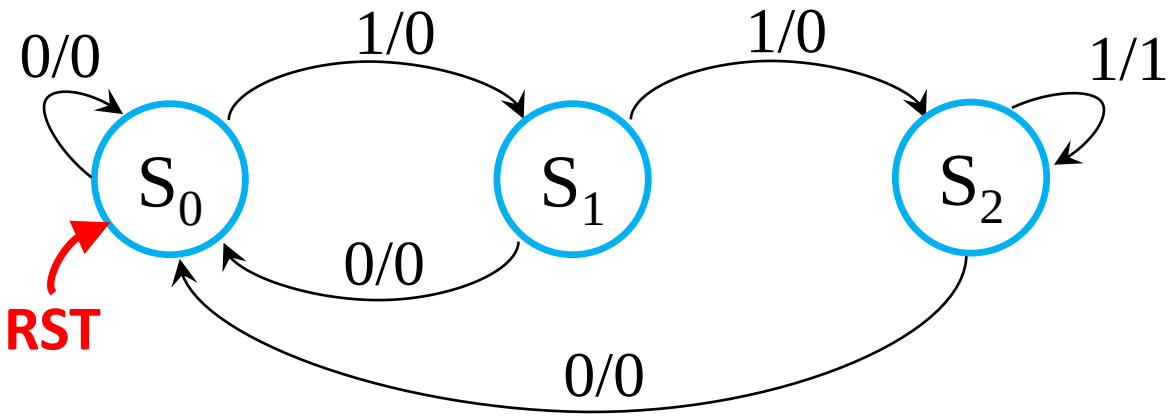
- Esempio di **costruzione del diagramma degli stati a partire dalle specifiche di un circuito**
- Un rilevatore di sequenza per comunicazioni seriali deve essere in grado di rilevare su una linea di dati in ingresso la presenza di tre 1 consecutivi. Quando l'ingresso X del circuito presenta tre 1 consecutivi, l'uscita Z diventa 1; in tutti gli altri casi, Z = 0. Una volta che Z = 1, vi rimane fintanto che non si presenta uno 0 in ingresso. Se X = 0, avviene un reset del sistema.

a) Disegnare il diagramma degli stati del circuito
b) Si tratta di una macchina di Mealy o di Moore?

Esercizio 4.26

- Ingresso X presenta tre 1 consecutivi => uscita Z = 1
- Altrimenti => uscita Z = 0
- Dopo che Z = 1, vi rimane fintanto che non arriva almeno uno 0
- Se X = 0, avviene un reset del sistema

a) Disegnare il **diagramma degli stati del circuito**

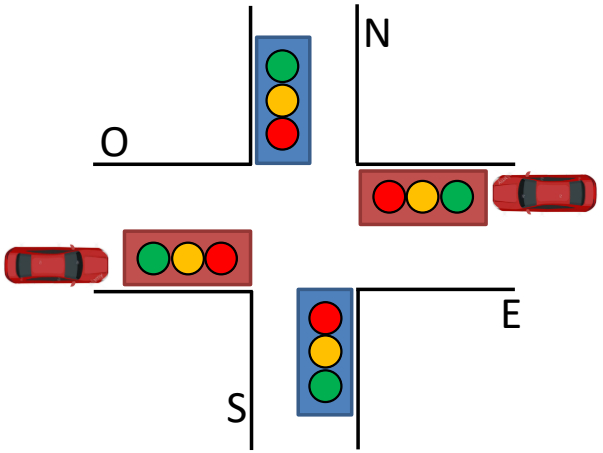


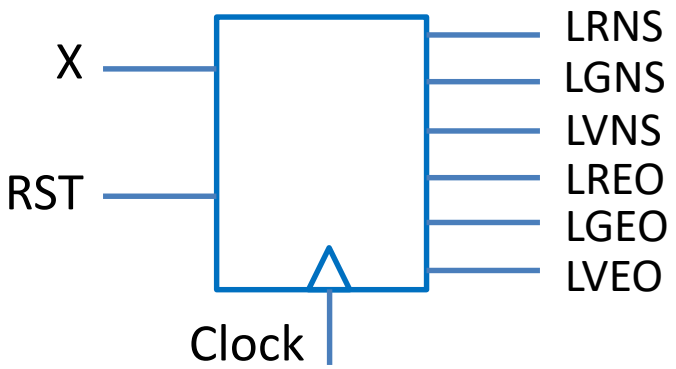
S0: nessun ‘1’ riconosciuto in ingresso
S1: un ‘1’ riconosciuto in ingresso
S2: due ‘1’ riconosciuti in ingresso

b) E’ una **macchina di Mealy**, infatti l’uscita dipende sia dallo stato che dall’ingresso

Controllore semaforico

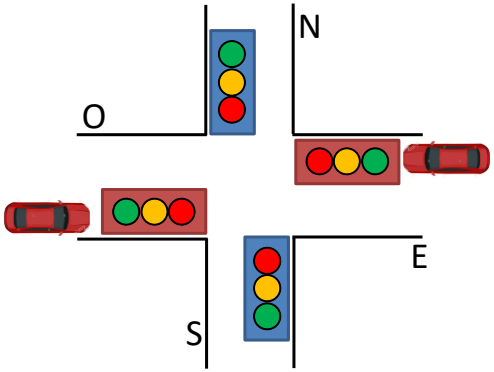
- Esempio di **sintesi di un circuito sequenziale sincrono**
- In un incrocio stradale all'intersezione tra una strada N-S con una strada E-O, ci sono 4 semafori. Progettare un circuito sequenziale per il controllo dei semafori, con in ingresso un segnale X che indica la presenza di un'auto in attesa in direzione E-O e in uscita 6 segnali che indicano se le corrispondenti luci dei semafori vanno accese (rosso N-S, giallo N-S, verde N-S, rosso E-O, giallo E-O, verde E-O) con le seguenti specifiche:
  - RESET: porta a semaforo verde in direzione N-S e rosso in direzione E-O
  - Se è rilevata un'auto in direzione E-O (X = 1), il sistema entra in una sequenza che fa diventare verde il semaforo E-O e poi fa tornare il verde nella direzione N-S
  - Ogni transizione al semaforo rosso deve essere preceduta da una transizione al giallo
  - Una direzione può avere il semaforo verde solo se nell'altra direzione è rosso

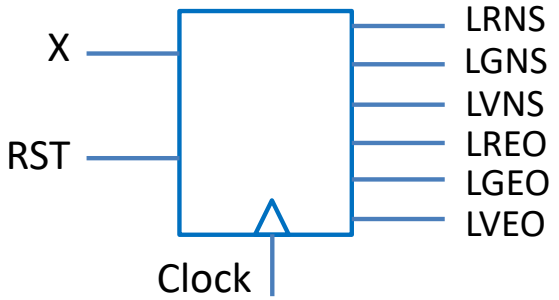


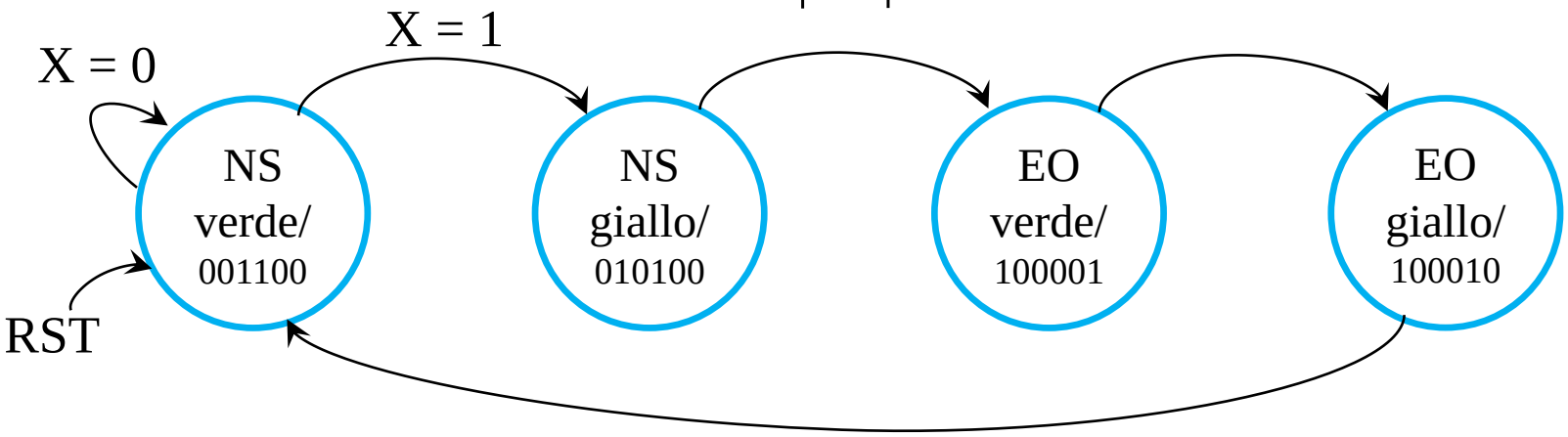


Controllore semaforico: diagramma degli stati

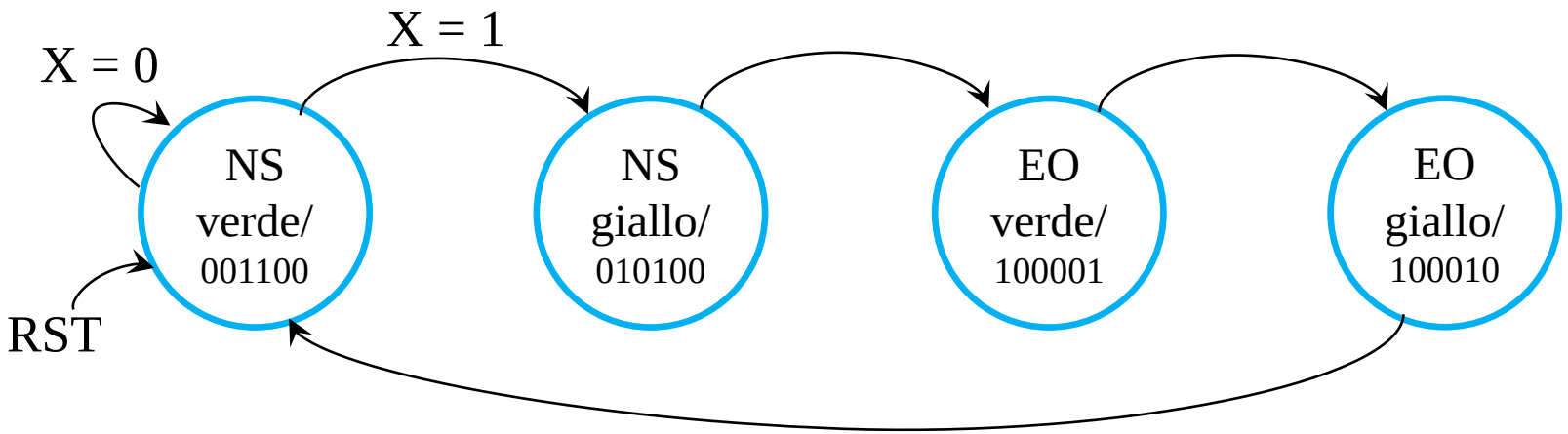
- RESET: porta a semaforo verde in direzione N-S e rosso in direzione E-O
- Se è rilevata un'auto in direzione E-O (X = 1), il sistema entra in una sequenza che fa diventare verde il semaforo E-O e poi fa tornare il verde nella direzione N-S
- Ogni transizione al semaforo rosso deve essere preceduta da una transizione al giallo
- Una direzione può avere il semaforo verde solo se nell'altra direzione è rosso

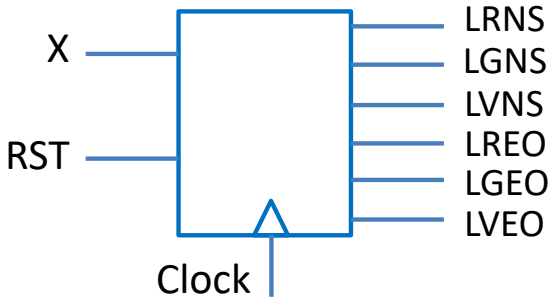


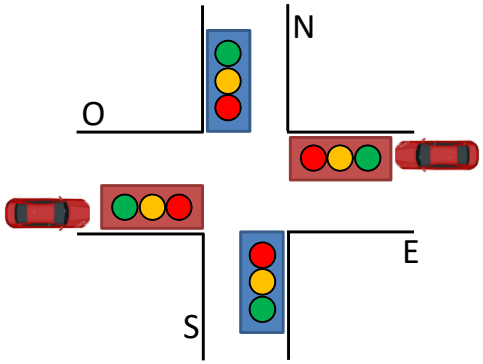




Controllore semaforico: tabella degli stati



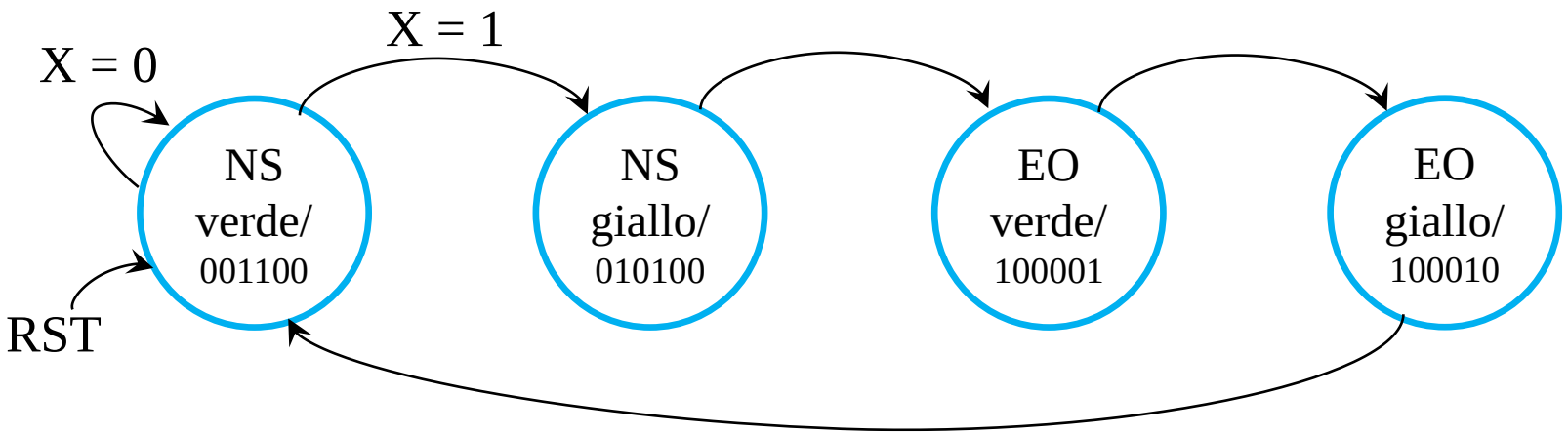


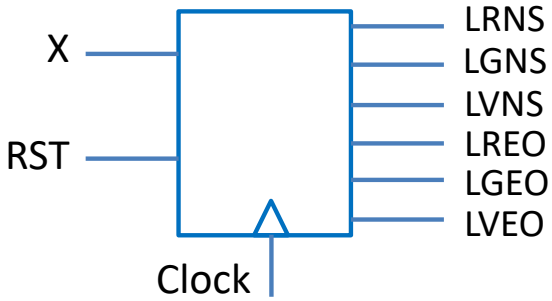


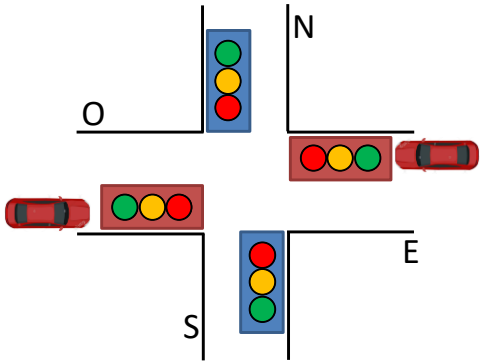
Stato pres	Stato futuro		Uscita					
	X=0	X=1	LRNS	LGNS	LVNS	LREO	LGEO	LVEO
VNS	VNS	GNS	0	0	1	1	0	0
GNS	VEO	VEO	0	1	0	1	0	0
VEO	GEO	GEO	1	0	0	0	0	1
GEO	VNS	VNS	1	0	0	0	1	0

=> Macchina di Moore: uscita dipende solo dallo stato

Controllore semaforico: codifica degli stati

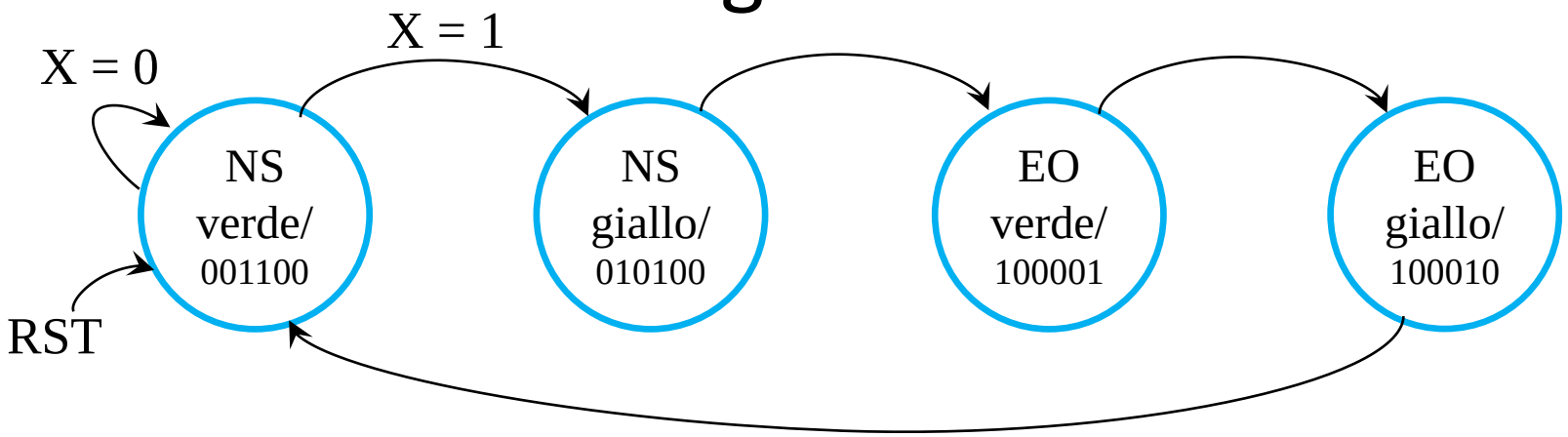






Stato	Codifica Gray AB
VNS	00
GNS	01
VEO	11
GEO	10

Controllore semaforico: tabella con codifica degli stati

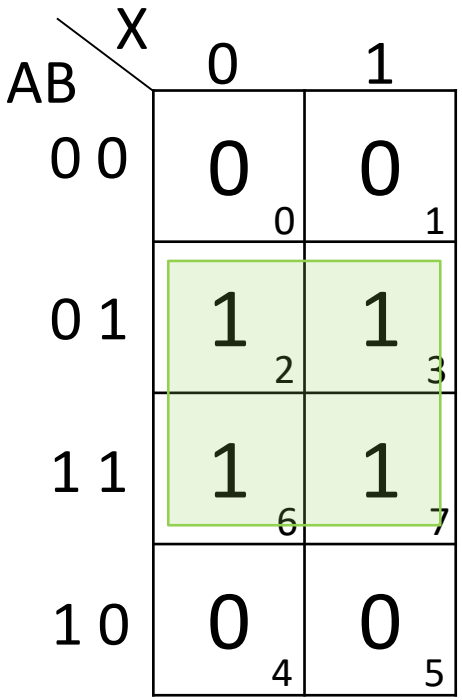


Stato	Codifica Gray AB
VNS	00
GNS	01
VEO	11
GEO	10

Stato pres AB	Stato futuro D_A D_B		Uscita LRNS LGNS LVNS LREO LGEO LVEO					
	X=0	X=1						
00	00	01	0	0	1	1	0	0
01	11	11	0	1	0	1	0	0
11	10	10	1	0	0	0	0	1
10	00	00	1	0	0	0	1	0

Controllore semaforico: equazioni dello stato futuro

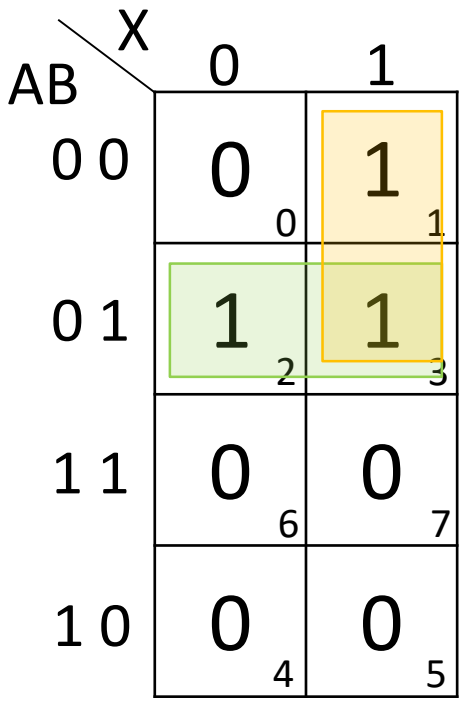
Stato presente AB	Stato futuro D_A D_B		Uscita
	X=0	X=1	
00	00	01	001 100
01	11	11	010 100
11	10	10	100 001
10	00	00	100 010



$$D_A = B$$

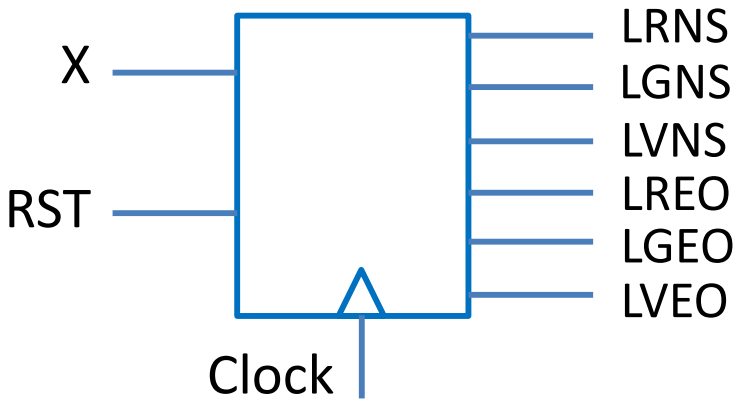
Controllore semaforico: equazioni dello stato futuro

Stato presente AB	Stato futuro D_A D_B		Uscita
	X=0	X=1	
00	00	01	001 100
01	11	11	010 100
11	10	10	100 001
10	00	00	100 010



$$D_B = \overline{A}B + \overline{A}X = \overline{A}(B + X)$$

Controllore semaforico: equazioni delle uscite



Stato presente AB	Stato futuro D_A D_B		Uscita
	X=0	X=1	
00	00	01	001 100
01	11	11	010 100
11	10	10	100 001
10	00	00	100 010

$$LRNS = A$$

$$LGNS = \overline{A} \cdot B$$

$$LVNS = \overline{A} \cdot \overline{B}$$

$$LREO = \overline{A}$$

$$LGEO = A \cdot \overline{B}$$

$$LVEO = A \cdot B$$

Controllore semaforico: circuito finale

Equazioni di aggiornamento dello stato:

$D_A = B$

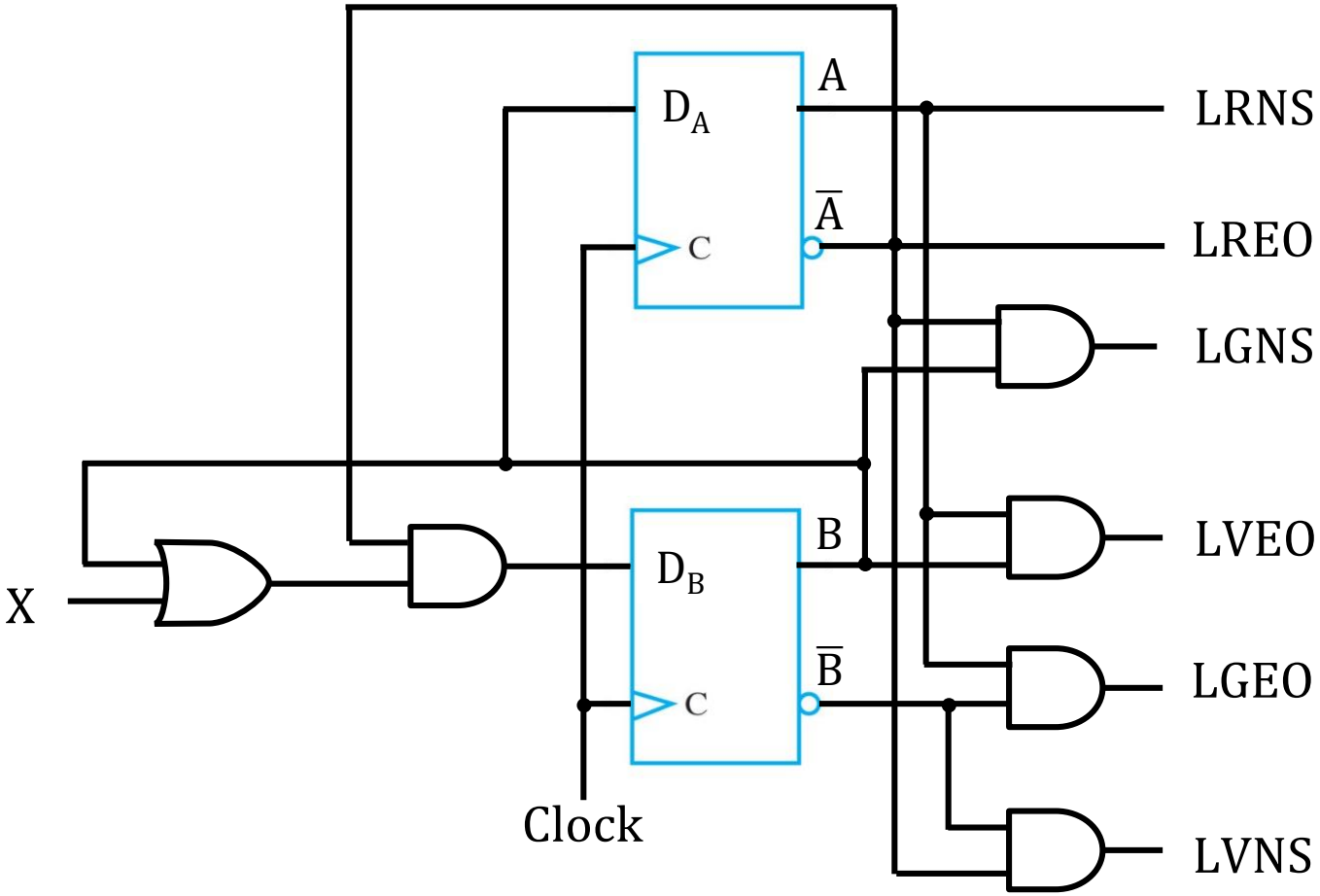
$D_B = \overline{A}B + \overline{A}X = \overline{A}(B + X)$

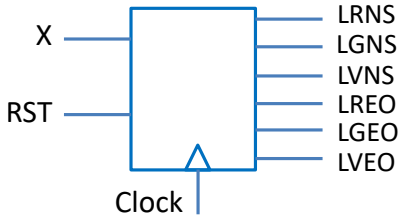
Equazioni delle uscite:

$LRNS = A$ $\quad LREO = \overline{A}$

$LGNS = \overline{A} \cdot B$ $\quad LGEO = A \cdot \overline{B}$

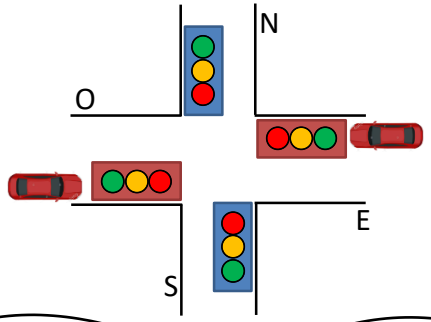
$LVNS = \overline{A} \cdot \overline{B}$ $\quad LVEO = A \cdot B$

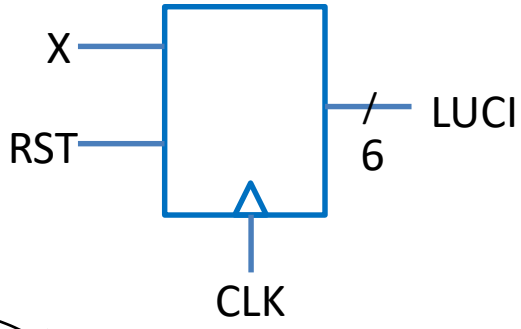


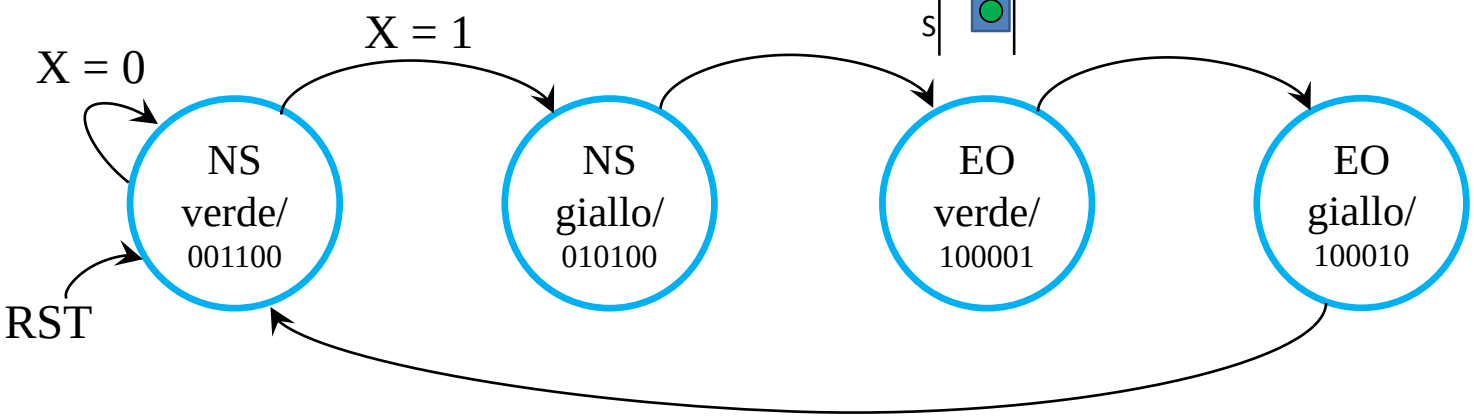


Controllore semaforico

- Realizzare una **descrizione VHDL behavioral** del circuito sequenziale sincrono per un controllo semaforico con le seguenti caratteristiche:
  - Ingresso X, Uscite RNS, GNS, VNS, REO, GEO, VEO
  - RESET: porta a semaforo verde in direzione N-S e rosso in direzione E-O
  - Se rileva un'auto in direzione E-O (X = 1), il sistema entra in una sequenza che fa diventare verde il semaforo E-O e poi fa tornare il verde nella direzione N-S
  - Prima di ogni transizione a semaforo rosso deve esserci una transizione al giallo
  - Una direzione può avere il semaforo verde solo se nell'altra direzione è rosso

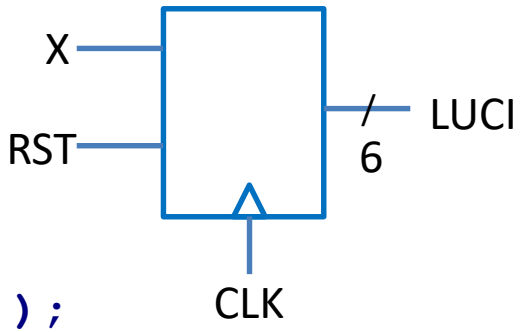






Controllore semaforico

```
library IEEE;
use IEEE.std_logic_1164.all;

entity semaforo is
   port ( clk, rst, X: in std_logic;
      luci: out std_logic_vector(5 downto 0));
-- (LRNS, LGNS, LVNS, LREO, LGEO, LVEO)
end semaforo;

architecture impl_beh of semaforo is

-- costanti per codifica dello stato
   subtype state_type is std_logic_vector(1 downto 0);
   constant VNS: state_type := "00";
   constant GNS: state_type := "01";
   constant VEO: state_type := "10";
   constant GEO: state_type := "11";
   signal state, next_state: state_type;
begin
```

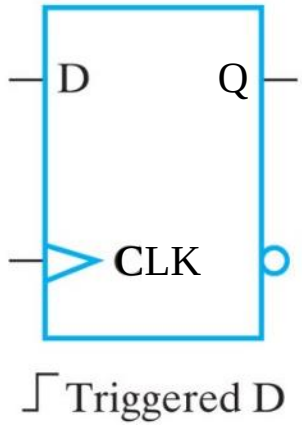


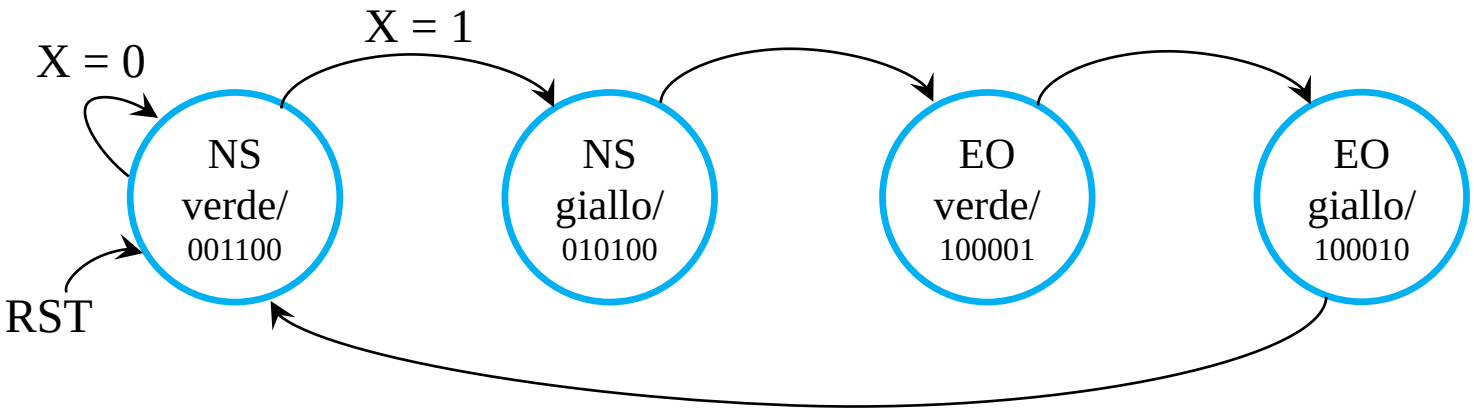
Vantaggioso usare delle costanti: se si vuole cambiare codifica, basta cambiare qui (il resto del codice rimane invariato)

Controllore semaforico

```
-- process 1: aggiorna lo stato e implementa il reset

   state_register: process (clk, rst)
      begin
         if (rst = '1') then
            state <= VNS;
         elsif (clk'event and clk = '1') then
            state <= next_state;
         end if;
    end process;
```

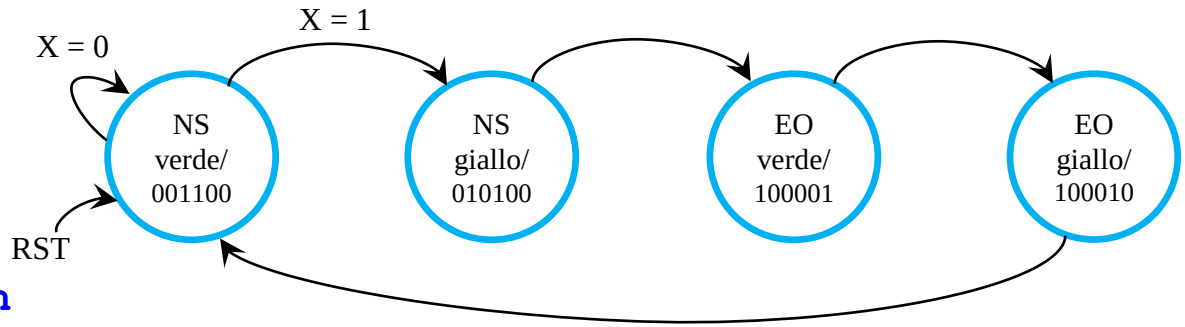
Processo 1: Realizza il registro di stato con D-ff positive edge triggered e implementa il RESET





Controllore semaforico

```
-- process 2: stato futuro come funzione di stato e ingresso
   next_state_comb: process (X, state)
      begin

      case state is

        when VNS =>
           if X = '0' then
             next_state <= VNS;
           else
             next_state <= GNS;
           end if;

        when GNS => next_state <= VEO;

        when VEO => next_state <= GEO;

        when GEO => next_state <= VNS;

        when others => null;
      end case;
   end process;
```



Processo 2: Aggiorna il valore dello stato

Bisogna enumerare tutte le possibili alternative, state è uno std_logic_vector!

Controllore semaforico

```
-- process 3: implementa l'uscita come funzione dello stato
-- LRNS, LGNS, LVNS, LREO, LGEO, LVEO

   output_comb: process (state)
   begin
      case state is

         when VNS => luci <= "001100";

         when GNS => luci <= "010100";

         when VEO => luci <= "100001";

         when GEO => luci <= "100010";

         when others => luci <= "XXXXXX";

      end case;
   end process;
end impl_beh;
```

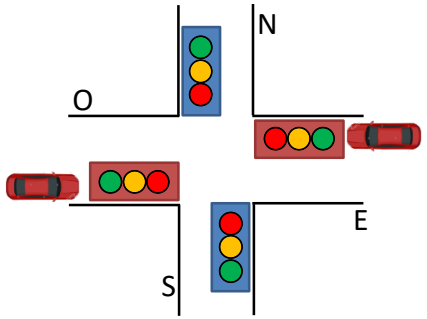
Processo 3: Calcola il valore dell'uscita

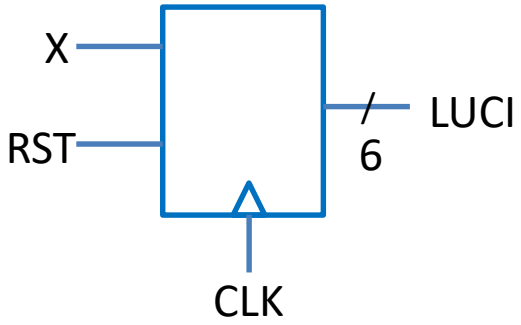
Stato	Uscita
VNS	001 100
GNS	010 100
VEO	100 001
GEO	100 010

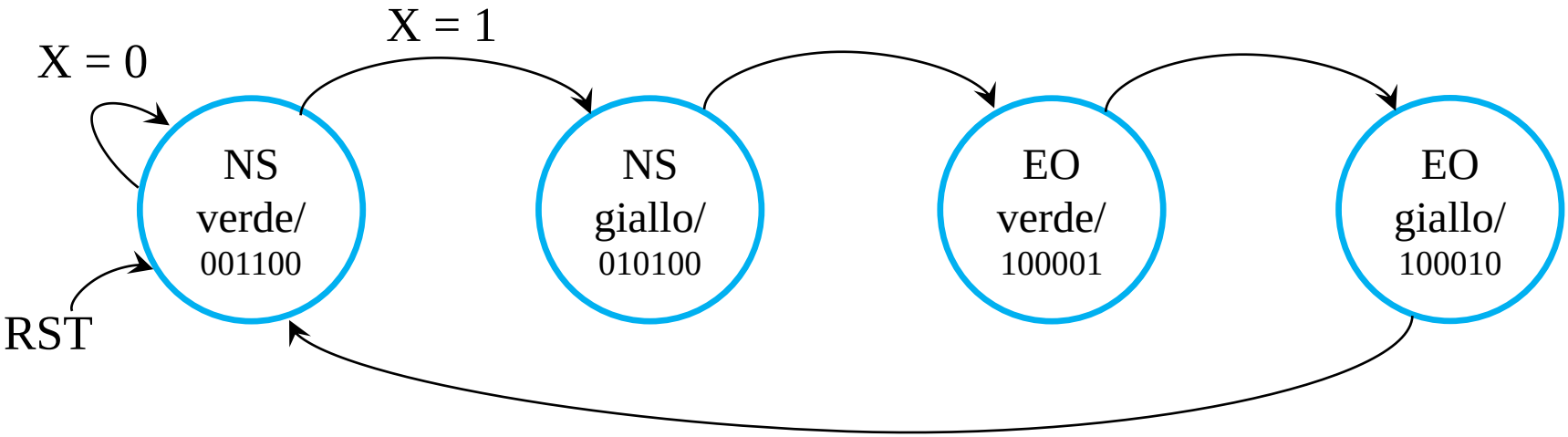
Se 'state' non assume uno dei valori possibili, l'uscita viene posta al valore 'undefined'

Controllore semaforico: testbench

- Realizzare un **testbench** del sistema per il controllo semaforico, applicando e simulare il circuito, in modo che tutti gli stati e tutte le transizioni della macchina vengano attraversati

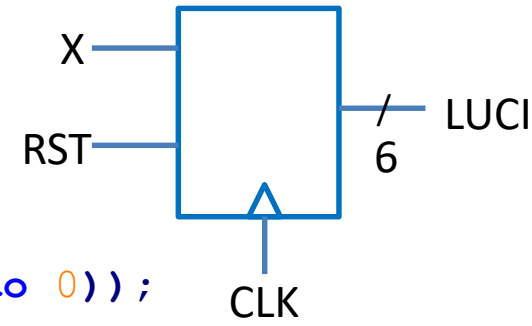






Controllore semaforico: testbench

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity test_semaforo is
end test_semaforo;

architecture test of test_semaforo is

signal clock, reset, autoEO: std_logic;
signal output: std_logic_vector (5 downto 0);

constant PERIOD: time := 100 ns;

component semaforo is
  port ( clk, rst, X: in std_logic;
                luci: out std_logic_vector(5 downto 0));
end component;

begin

DUT: semaforo port map(clock, reset, autoEO, output);
```



Controllore semaforico: testbench

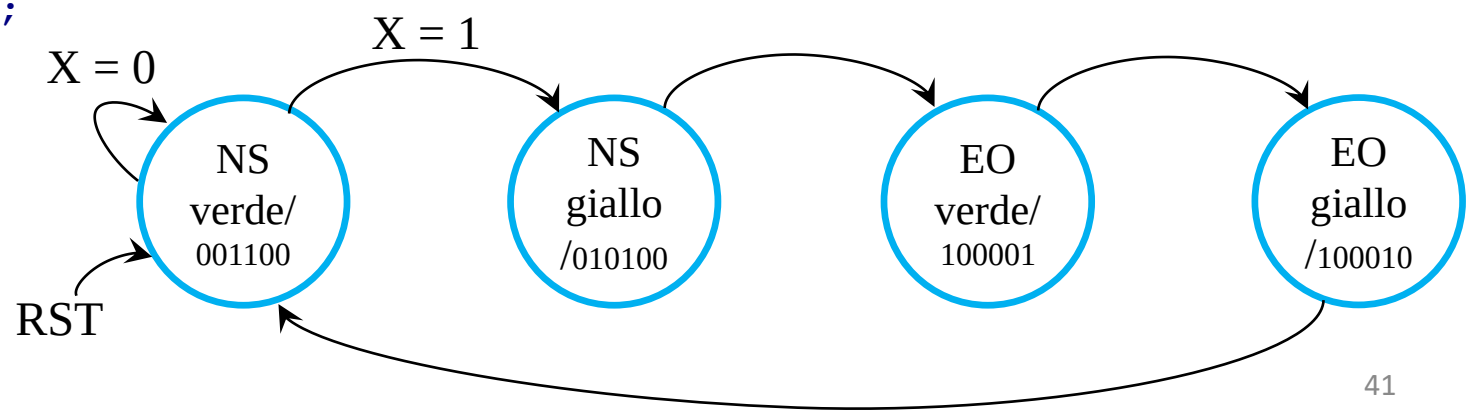
```
generate_clock: process
   begin
      clock <= '1';
      wait for PERIOD/2;
      clock <= '0';
      wait for PERIOD/2;
end process;
```

Genera un clock con periodo PERIOD: processo viene eseguito di continuo

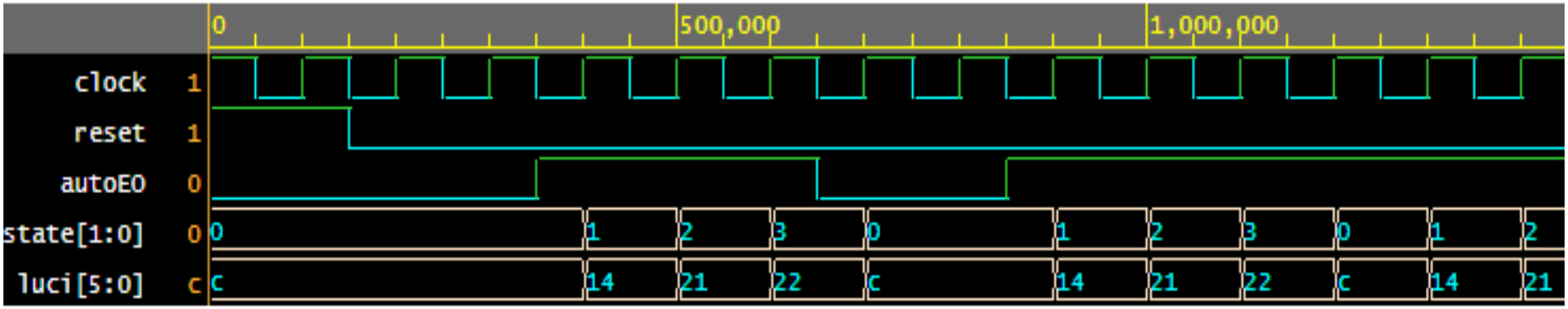
Controllore semaforico: testbench

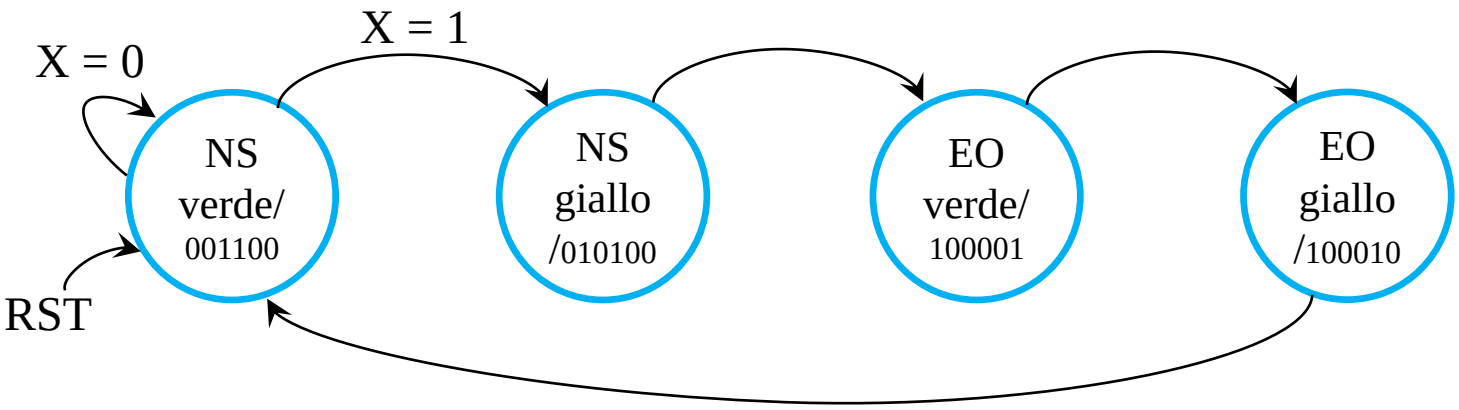
```
apply_inputs: process
   begin
       reset <= '1';
       autoEO <= '0';
       wait for 3*PERIOD/2;
       reset <= '0';
       wait for 2*PERIOD;
       autoEO <= '1';      -- aspetta 2 cicli di clk, poi arriva un'auto
       wait for 3*PERIOD;
       autoEO <= '0';      -- auto riparte dopo 3 cicli di clk
       wait for 2*PERIOD;
       autoEO <= '1';      -- dopo 2 cicli di clk arriva un'altra auto
       wait for 6*PERIOD;
       std.env.stop;
   end process;

end test;
```

Applica gli input al circuito (al fronte di discesa del clk)



Controllore semaforico: forme d'onda





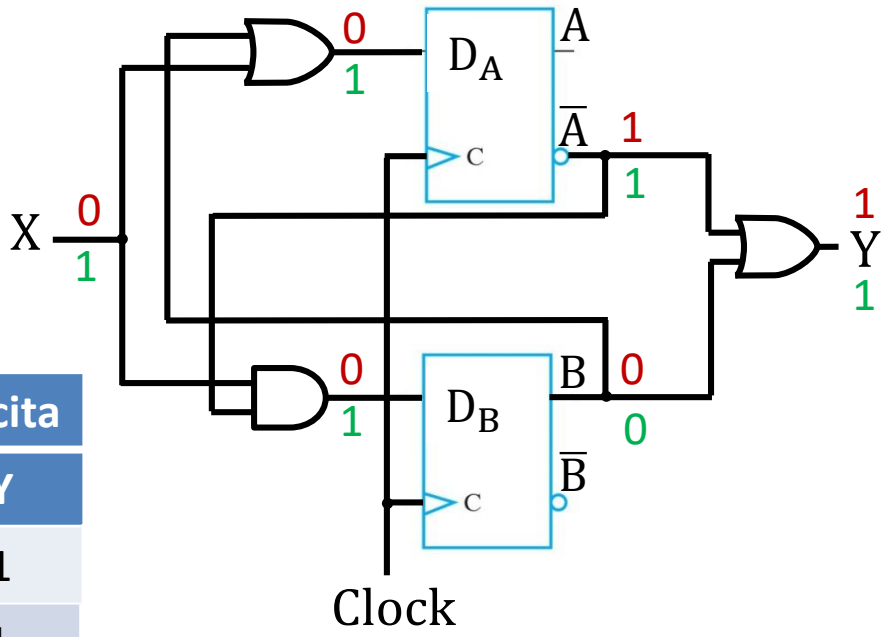
Disclaimer

Figures from *Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano | Kime | Martin

Esercizio 4.6

b) Derivare la **tabella degli stati**

- $Y = \overline{A} + B$
- $D_A = X + B$
- $D_B = X \cdot \overline{A}$



Stato presente		Ingresso	Stato futuro		Uscita
A	B	X	D_A	D_B	Y
0	0	0	0	0	1
0	0	1	1	1	1